

1198-3878
US

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

1999年 2月19日

出 願 番 号

Application Number:

平成11年特許願第041032号

出 願 人

Applicant(s):

株式会社日立製作所

JC678 U.S. PRO
09/495897

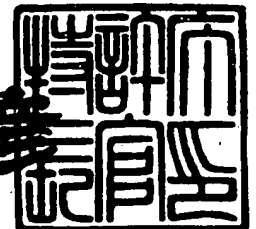


CERTIFIED COPY OF
PRIORITY DOCUMENT

1999年11月 5日

特許庁長官
Commissioner,
Patent Office

近 藤 隆 彦



【書類名】 特許願

【整理番号】 1198038781

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 9/46

【発明の名称】 複数のオペレーティングシステムを実行する計算機

【請求項の数】 10

【発明者】

 【住所又は居所】 茨城県日立市大みか町七丁目 1 番 1 号
株式会社 日立製作所 日立研究所内

 【氏名】 齊藤 雅彦

【発明者】

 【住所又は居所】 茨城県日立市大みか町七丁目 1 番 1 号
株式会社 日立製作所 日立研究所内

 【氏名】 上脇 正

【発明者】

 【住所又は居所】 茨城県日立市大みか町五丁目 2 番 1 号
株式会社 日立製作所 大みか工場内

 【氏名】 中村 智明

【発明者】

 【住所又は居所】 茨城県日立市大みか町五丁目 2 番 1 号
株式会社 日立製作所 大みか工場内

 【氏名】 大野 洋

【発明者】

 【住所又は居所】 神奈川県川崎市麻生区王禅寺 1 0 9 9 番地
株式会社 日立製作所 システム開発研究所内

 【氏名】 井上 太郎

【特許出願人】

 【識別番号】 000005108

 【氏名又は名称】 株式会社 日立製作所

【代理人】

【識別番号】 100068504

【弁理士】

【氏名又は名称】 小川 勝男

【電話番号】 03-3212-1111

【手数料の表示】

【予納台帳番号】 013088

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 複数のオペレーティングシステムを実行する計算機

【特許請求の範囲】

【請求項 1】

複数のオペレーティングシステムと、それぞれの上記オペレーティングシステムのプログラム上で実行される複数のプロセスまたはスレッドとを記憶したメモリと、

それぞれの上記プロセス又はスレッドに割り当てられた優先順位に基づいて、オペレーティングシステムを実行する処理装置とを有する計算機であって、

上記処理装置は、それぞれの上記オペレーティングシステムで実行すべきプロセス又はスレッドの優先順位を読み出し、それぞれ読み出された優先順位を複数の上記オペレーティングシステム間で共通の優先順位に変換し、上記変換結果に基づいて優先して実行すべきオペレーティングシステムを決定するとともに、上記決定されたオペレーティングシステムを実行する計算機。

【請求項 2】

請求項 1 において、上記メモリは複数の上記オペレーティングシステムで実行すべきプロセス又はスレッドの優先順位を上記共通の優先順位にマッピングするための優先順位変換テーブルを有し、上記プロセッサは上記変換テーブルに基づいて優先して実行すべきオペレーティングシステムを決定する計算機。

【請求項 3】

請求項 1 又は 2 において、上記プロセッサは、複数のオペレーティングシステム間で共通の優先順位から、それぞれの上記オペレーティングシステムにおける優先順位を決定し、それぞれのオペレーティングシステムで実行される複数の上記プロセス又はスレッドの優先順位を変更する計算機。

【請求項 4】

請求項 3 において、上記メモリは、上記共通の優先順位をそれぞれの上記オペレーティングシステムにおける優先順位にマッピングするための優先順位逆変換テーブルを有し、上記プロセッサは、上記優先順位逆変換テーブルに基づいて複数の上記プロセス又はスレッドの優先順位を変更する計算機。

【請求項 5】

請求項 1 乃至 4 において、上記プロセッサはプロセス又はスレッドが指定された処理を実行する場合に、該プロセス又はスレッドを実行するオペレーティングシステムの優先順位を上昇させ、上記プロセス又はスレッドが該指定された処理を終了した場合に、上記オペレーティングシステムの優先順位を下降させる計算機。

【請求項 6】

複数のプロセスまたはスレッドと、

複数の上記プロセス又はスレッドを実行すると共に、実行しているプロセス又はスレッドの優先順位を通知する複数のオペレーティングシステムと、

それぞれの上記オペレーティングシステムから通知された優先順位をオペレーティングシステム間で共通の優先順位に変換する優先順位変換処理と、

上記優先順位変換処理によって得られた共通の優先順位を比較して、より高い共通の優先順位を有するオペレーティングシステムを優先的に実行させる優先順位比較処理とを記憶した情報記憶媒体。

【請求項 7】

複数のオペレーティングシステムを選択的に実行するオペレーティングシステム実行方法であって、

優先順位変換処理によって、それぞれの上記オペレーティングシステムで実行されるプロセス又はスレッドの優先順位をオペレーティングシステム間で共通の優先順位に変換し、

優先順位比較処理によって、上記優先順位変換処理によって得られた共通の優先順位を比較して、より高い共通の優先順位を有するオペレーティングシステムを優先的に実行させるオペレーティングシステム実行方法。

【請求項 8】

請求項 7 において、上記優先順位変換処理はそれぞれのオペレーティングシステムの優先順位を上記共通の優先順位に変換するにあたって、異なったオペレーティングシステムの優先順位は互いに異なった共通の優先順位に変換するオペレーティングシステム実行方法。

【請求項 9】

請求項 7 又は 8 において、前記優先順位変換処理は、それぞれの上記オペレーティングシステムが実行しているプロセスまたはスレッドの優先順位を上記共通の優先順位に変換するほかに、少なくとも、割込み処理中の状態、オペレーティングシステム自体の機能を実行中の状態、実行すべき処理が存在しない状態という三つの状態を共通の優先順位に変換するオペレーティングシステム実行方法。

【請求項 10】

複数のプロセスまたはスレッドを該複数のプロセスまたはスレッドに割り当てられた優先順位にしたがって実行する複数個のオペレーティングシステムと、該複数個のオペレーティングシステムを切替える手段とを有する計算機システムにおいて、該複数個のオペレーティングシステムは、それぞれが実行しているプロセスまたはスレッドの優先順位を計算機システム内で共通の優先順位に変換する優先順位変換手段と、該優先順位変換手段から得られた共通の優先順位を該オペレーティングシステムに切替える手段に通知する優先順位通知手段とを有し、該オペレーティングシステムを切替える手段は、それぞれのオペレーティングシステムから通知された共通の優先順位を比較して、より高い共通の優先順位を有するオペレーティングシステムを優先的に実行させる優先順位比較手段を有したことを特徴とする計算機システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は複数個のオペレーティングシステムを切り替えつつ動作させる計算機およびそのオペレーティングシステム切り替え方式を対象分野とし、特に、相異なる優先順位体系を有するオペレーティングシステムの切替え方式に関連する。

【0002】

【従来の技術】

一つの計算機で複数のオペレーティングシステムを動作させる技術として、従来から大型計算機において、「仮想計算機(Virtual Machine: VM)」が知られている。該従来技術では、計算機内で並行して動作する複数個の仮想計算機それ

それぞれの上で複数のユーザタスク（以降、プロセスおよびスレッドを統一してタスクと呼ぶ）を切り替えて処理を実行する。仮想計算機は、通常、大型計算機内の一つのプロセスとして実現されるが、仮想計算機とユーザタスクとの関連を考慮すると、一つのオペレーティングシステムであると見なすことも可能である。

【0003】

一般に、それぞれの仮想計算機には、該仮想計算機の優先順位にしたがって一定のタイムスライス（CPU割当て時間）を与える。各仮想計算機は、与えられたタイムスライスの範囲内でユーザタスクを切り替えて動作させる。このような仮想計算機技術の実行効率を高める技術として、特開平5-197577号に開示された「仮想計算機における仮想計算機実行プライオリティ制御方式」がある。

【0004】

該従来技術は、複数の仮想計算機と、該複数の仮想計算機を制御するための仮想計算機モニタとから構成される。仮想計算機は、システムタスクのように実行優先順位の高いタスクの実行開始・実行終了時に、仮想計算機モニタに対してタスクの優先順位を通知する。これを受けた仮想計算機モニタは、仮想計算機の実行優先順位を、通知された優先順位に変更する。仮想計算機の実行優先順位を仮想計算機内で実行されるタスクの優先順位に変更することにより、仮想計算機制御を効率的に行うことができるとしている。

【0005】

【発明が解決しようとする課題】

マイクロプロセッサ、特に、組込み向けマイクロプロセッサの性能向上と、オペレーティングシステムの機能向上に伴って、複数の相違なるオペレーティングシステムを一つの計算機上で同時に動作させ、これらを動的に切り替えて処理を行うというユーザニーズが現われている。

【0006】

一般に、工場・プラントなどの機械制御、車載ナビゲーションシステムといった分野では、外部環境変化に対して即座に応答するというリアルタイム性、長時間連続運転を可能とする信頼性などが重要視されている。このため、割込み応答性が高く、かつ、コンパクトでモジュール構成を有するリアルタイム用オペレー

ティングシステム（リアルタイムOS）が用いられることが多い。しかしながら、リアルタイムOSは、リアルタイム性・信頼性を重要視する反面、人間とのインタフェースに優れているとは言い難い。

【0007】

一方、一般のパソコン（PC）に用いられている事務処理用オペレーティングシステム（事務処理OS）は、画像を用いて操作を行えるようにするなど、対人間インタフェースに優れている。このため、従来はリアルタイムOSを使用していた分野にも事務処理OSのユーザインタフェースを使用したいという要求が高まっている。しかし、事務処理OSは人間との対話を主な処理としているため、割込み応答性よりも処理のスループットを重要視し、比較的長時間割込み禁止状態のまま処理を実行することもある。また、コンパクトな構成を有するリアルタイムOSに対し、信頼性において匹敵するとは言い難く、24時間連続運転などに向いていないという側面もある。

【0008】

しかしながら、大型計算機上で複数の仮想計算機（オペレーティングシステム）を並行して動作させる方式と同様に、同一計算機上で事務処理OSとリアルタイムOSとを動作させ、必要に応じてこれらのオペレーティングシステムを切り替えることができれば、優れたユーザインタフェースとリアルタイム性・信頼性とを両立させることができる。マイクロプロセッサの性能向上を考慮すれば、複数のオペレーティングシステムを一つの計算機上で動作させることは、大型計算機だけに許された技術ではなくなっている。

【0009】

このとき、それぞれのオペレーティングシステムの重要性を考慮すると、常にリアルタイムOSを優先的に動作させ、リアルタイムOS上で実行させるタスクが存在しなくなった場合にのみ、事務処理OSを動作させるという方法が最も単純なオペレーティングシステム切替え方式である。しかしながら、各オペレーティングシステム上で動作する個々のタスクの重要性が、このように（リアルタイムOS上のタスクは、常に、事務処理OS上のタスクよりも重要性が高いというように）単純に切り分けられるとは限らない。

【0 0 1 0】

図 2 7 に単純な切り分けを行えない例を示す。図 2 7 は車載ナビゲーションシステムの構成例を示したものである。ここでは、システムを簡略化し、車載ナビゲーションシステムが、（１）運転位置を認識する位置認識タスク 3 7 0、（２）目標地点までの最短経路を導出する経路探索タスク 3 7 1、（３）ボタン・タッチパネルなどからの入力进行处理するインタフェースタスク 3 7 2、（４）運転休憩中に起動していたゲームタスク 3 7 3、という四つのタスクから構成されていると仮定する。通常、高速応答性・信頼性が求められる位置認識タスク・経路探索タスクはリアルタイム OS 1 1 1 上で実行され、ユーザインタフェースに優れた事務処理 OS 1 1 0 では、インタフェースタスク・ゲームタスクなどを実行する。しかしながら、一般に、経路探索は非常に計算量の大きい処理であり、数秒オーダの計算時間を要することもある。上記に示した単純な重要性の切り分けを行うと、この間、インタフェースタスクの処理は停止し、ユーザがいくらボタンを押しても認識されないという問題が発生する。

【0 0 1 1】

図 2 7 に示した車載ナビゲーションシステムではインタフェースタスクの重要性が高く、これが実行可能状態に移行した場合には、事務処理 OS を優先的に実行する必要がある。しかしながら、前記従来技術「仮想計算機における仮想計算機実行プライオリティ制御方式」は、各仮想計算機（オペレーティングシステム）の機能が同一であることを前提としている。一般に、リアルタイム OS は環境変化に高速に応答するため、事務処理 OS に比べて極めて多くの優先順位レベルを有することが多い。また、極端な場合、リアルタイム OS 上では、優先順位の

「数値」が小さい（０に近い）ほど優先順位が高く、事務処理 OS 上では、優先順位の「数値」が大きいほど優先順位が高い、といった逆の意味付けがなされていることもある。このような場合、たとえ二つのオペレーティングシステムから優先順位を通知されても、いずれのオペレーティングシステムを優先的に起動してよいか判断できないことになる。このため、前記従来技術では、異なった機能を有する事務処理 OS とリアルタイム OS とを統一して管理することはできない。

【0 0 1 2】

本発明の目的は、複数の相異なるオペレーティングシステムを一つの計算機上で動作させるマルチオペレーティングシステム制御装置において、各オペレーティングシステム上で実行されているタスクの重要性を考慮して、動作させるべきオペレーティングシステムを選択・切り替えることであり、これによって、重要性のあるタスクが優先的に実行されるようにすることである。

【0 0 1 3】

【課題を解決するための手段】

上記目的を達成するため、本発明では、複数のタスクをそれぞれのタスクに割当てられた優先順位にしたがって実行する複数のオペレーティングシステムと、この複数のオペレーティングシステムを切り替えて実行する計算機において、下記に示す処理を行う。

【0 0 1 4】

- (1) 各オペレーティングシステムが実行しているタスクの優先順位を確認する優先順位監視処理を行う。若しくは、各オペレーティングシステム内に、自分が実行しているタスクの優先順位を通知する優先順位通知処理を行う。事務処理OS, リアルタイムOSでは、内部に優先順位通知処理を追加することが不可能なこともある。このとき、前者の優先順位監視処理行う必要がある。

【0 0 1 5】

- (2) 各オペレーティングシステムから入手した実行中タスクの優先順位を、それぞれのオペレーティングシステムで共通の優先順位に変換する優先順位変換処理を行う(以降、該共通の優先順位のことを「正規化優先順位」と呼ぶ。)
- (3) 優先順位変換処理によって得られた各オペレーティングシステムの正規化優先順位を比較して、動作させるべきオペレーティングシステムを決定し、かつ、オペレーティングシステムの切り替えを実行する優先順位比較処理を行う。

【0 0 1 6】

本発明によれば、優先順位通知処理は各オペレーティングシステム内に設けられ、オペレーティングシステムがタスク切り替えを行うたびに、該タスクの優先

順位を通知する。

【0017】

タスク切り替え機能はオペレーティングシステムの中核を構成するものであるため、市販されているオペレーティングシステム内に優先順位通知手段を組み込むことが不可能なことも多い。しかしながら、一般に、オペレーティングシステムは実行中タスクの管理情報を計算機のメモリ上に保持し、この管理情報の一部にタスクの実行優先順位がある。また、タスク切り替え処理を高速化するため、現在実行中のタスクの優先順位を特定の変数（優先順位保持変数）に格納しておく場合もある。優先順位監視処理は、タスク管理情報または優先順位保持変数を読み出して、各オペレーティングシステムが実行しているタスクの優先順位を確認する。優先順位監視処理は、外部割込み、タイマ割込みなどのタイミングで優先順位の確認を行う。

【0018】

優先順位変換処理は、各オペレーティングシステムから入手した実行中タスクの優先順位を、計算機内で共通の正規化優先順位に変換する。このため、優先順位変換処理は、各オペレーティングシステムに対応する優先順位変換テーブルを有することがある。優先順位変換テーブルは、各オペレーティングシステム固有の優先順位から正規化優先順位を取り出すための対応表である。簡単な数式によって個々のオペレーティングシステムの優先順位を正規化優先順位に変換することも可能であるが、高速かつ柔軟なオペレーティングシステム切り替えを実行するためには、優先順位変換テーブルを用いるほうがよい。例えば、優先順位変換テーブルに格納する正規化優先順位の値を適切に定めることにより、個々のオペレーティングシステムから得られる正規化優先順位が互いに等しくならないように変換を行うことも可能である。

【0019】

優先順位比較処理は、各オペレーティングシステムの正規化優先順位を優先順位変換処理から入手し、現在動作させているオペレーティングシステムより高い正規化優先順位を有するオペレーティングシステムが存在すれば、オペレーティングシステムの切り替えを行う。

【0020】

【発明の実施の形態】

本発明の実施例を図面を用いて説明する。

【0021】

図1に本発明の第一の実施例における全体構成を示す。通常、計算機は、プロセッサ100、メモリ101、入出力制御装置102、ディスク装置104、ディスプレイ105などから構成される。プロセッサ100、メモリ101、入出力制御装置102はプロセッサバス103によって接続される。プロセッサ100は複数のオペレーティングシステムを動作させるためのマイクロプロセッサである。メモリ101は、オペレーティングシステムである事務処理OS110、リアルタイムOS111、各オペレーティングシステム上で動作するタスク112～117、オペレーティングシステム切り替えプログラム118を記憶する。これらのプログラムはプロセッサ100によって読み出されて実行される。

【0022】

入出力制御装置102には、プログラム・データを保存するためのディスク装置104、画面表示用デバイスであるディスプレイ105が接続される。また、工場・プラント制御用、あるいは、組込み向け計算機を実現する場合、入出力制御装置102には、リアルタイム制御ネットワーク106が接続されることがある。リアルタイム制御ネットワーク106には、センサ、アクチュエータなどの入出力機器が接続される。なお、入出力制御装置102に接続されるディスク装置104、ディスプレイ105、リアルタイム制御ネットワーク106の入出力装置104～106のうち、いずれかまたは全ては、システム構成によって省略されることもある。入出力制御装置102は、割込み信号線107によってプロセッサ100と接続され、入出力動作完了などを通知することができる。図1中では、説明のため、割込み信号線107とプロセッサバス103が別の装置であるように記載しているが、実際には、割込み信号線107はプロセッサバス103の一部である。プロセッサ100内部にはタイマ装置108が設けられており、一定周期毎に内部割込みを発生させる。タイマ装置108からの割込みは、オペレーティングシステムの計時などに使用される。

【0023】

プロセッサ100は割込み信号線107によって通知される外部割込みやタイマ装置108などからの内部割込みをマスクできる機能を装備する。割込みマスクとは、プログラムが割込みマスクを解除するまで、特定の割込みが入ることを遅延させる機能である。一般に、割込みマスク機能には、下記の三種類が存在する。

【0024】

- (1) 全割込みマスク：全ての割込みをマスクする。

【0025】

- (2) 個別割込みマスク：個々の割込みをそれぞれマスクできるようにする。

【0026】

- (3) 割込みレベルマスク：各割込みにレベルを設定し、指定レベル以下の割込みをマスクする。

【0027】

プロセッサ100の種類によって、上記(1)(2)の組合せ、または、(1)(3)の組合せのいずれかで構成することが多い。後者の組合せで構成するプロセッサを採用する場合、対応する入出力装置の重要性にしたがって割込みレベルを割当てることになる。例えば、リアルタイム制御ネットワーク106からの割込みを、ディスク装置104、ディスプレイ105などからの割込みより高いレベルに設定する。

【0028】

本実施例では、計算機内に二つのオペレーティングシステムである事務処理OS110、リアルタイムOS111が存在する。このオペレーティングシステムは、各自に割当てられたメモリ・プロセッサ資源を用い、それぞれ、タスク112～114、タスク115～117を実行させる。ここでは、オペレーティングシステム数が2個、タスク数が6個（各オペレーティングシステムに対して3個ずつ）の例を示しているが、これらの数値よりも多い、または、少ないオペレーティングシステム、タスクを実装することも可能である。本実施例では、オペレーティングシステム数の動的な変更は想定しないが、各オペレーティングシ

システムが動的にタスク生成・削除を行うことは可能である。また、本実施例では、事務処理OS 110, リアルタイムOS 111について説明するが、実際に、各オペレーティングシステムがいかなる種類であっても、本発明で述べる技術は適用可能である。タスク 112~114 は事務処理OS で実行される事務処理タスクであり、タスク 115~117 はリアルタイムOS で実行されるリアルタイムタスクである。事務処理OS 110 とリアルタイムOS 111 は、各々独立にタスクの優先順位を定義している。図 1 に示す実施例では、事務処理OS 110 環境下で、タスク 112~114 が、それぞれ、優先順位 0, 7, 31 を有し、リアルタイムOS 111 環境下で、タスク 115~117 が、それぞれ、優先順位 0, 99, 255 を有する。尚、本実施例においてタスクとはプロセス又はスレッドの総称を意味する。プロセスとは、個別のメモリ空間を有して動作するプログラムを意味する。このため、通常あるプロセスは別のプロセスのデータを変更することはできない。スレッドとは、プロセスのメモリ空間を共有して動作するプログラムを指す。従って、同一プロセス内で動作するスレッド同士の間ではデータ保護機能は存在しない。

【0029】

各オペレーティングシステム内には、優先順位通知モジュール 120, 121 が存在する。優先順位通知モジュール 120, 121 は、各オペレーティングシステムのタスク切り替え時に実行され、オペレーティングシステム切り替えプログラム 118 に対して次に実行すべきタスクの優先順位を通知する。

【0030】

一般に、事務処理OS とリアルタイムOS とは異なった優先順位体系を有することが多い。リアルタイムOS では、比較的多くの優先順位レベルを有することにより、重要な割込みに対する高速応答性を実現する。一方、事務処理OS では、比較的少ない優先順位レベルを用いて、スループットを向上させる方式を採用する。また、オペレーティングシステムの種類によっては、優先順位の数値が小さい（0に近い）ほど優先順位が高い場合と、優先順位の数値が大きいほど優先順位が高い場合とがある。このため、優先順位通知モジュール 120, 121 から得られた優先順位をそのまま比較しても意味がない。なお、以降、事務処理

OS 110では、優先順位の数値が大きいほどタスクの優先順位が高く（図1中では、タスク114が最も優先される）、リアルタイムOS 111では、逆に、優先順位の数値が小さいほどタスクの優先順位が高い（図1中では、タスク115が最も優先される）ものとして説明する。また、事務処理OS 110における優先順位は0～31の範囲で指定でき、リアルタイムOS 111における優先順位は0～255の範囲で指定できるものとする。

【0031】

二つのオペレーティングシステムにおける優先順位体系の違いを解消するため、オペレーティングシステム切り替えプログラム118は、優先順位変換モジュール122、123と優先順位比較モジュール124とを構成要素として有する。優先順位変換モジュール122、123は、それぞれ、事務処理OS 110、リアルタイムOS 111から得られた優先順位をシステム内で共通の比較基準である正規化優先順位に変換する。各オペレーティングシステムの優先順位をそのまま比較することは不可能であるが、一旦両者を正規化優先順位に変換すれば、両オペレーティングシステムが実行しているタスクの優先順位を比較することができる。なお、ここで、正規化優先順位が一方のオペレーティングシステムの優先順位と全く同一であってもよい。この場合、当該オペレーティングシステムの優先順位がそのまま正規化優先順位となる。優先順位比較モジュール124は、両オペレーティングシステムから得られた正規化優先順位を比較し、かつ、より高い正規化優先順位を有するオペレーティングシステムに切り替えて実行させる役割を有する。

【0032】

図2に、プロセッサ100の内部構成例を示す。キャッシュメモリ131はメモリ101上のデータまたは命令を一時的に格納するバッファ記憶装置である。CPU130は演算回路であり、メモリ101若しくはキャッシュメモリ131上に存在する命令を順次読み出して実行する。命令実行に際して、演算結果を一時的に保持するための汎用レジスタ132、命令アドレスを指定するプログラムカウンタ133、実行状態を保持するステータスレジスタ134を用いる。CPU130、キャッシュメモリ131、汎用レジスタ132、プログラムカウンタ133、ステ

ータスレジスタ134は、互いに、データ転送を行う複数の信号線であるデータバス136，アドレス指定を行う複数の信号線であるアドレスバス137によって接続されている。

【0033】

割込み信号線107とタイマ装置108は割込みコントローラ135に接続される。割込みコントローラ135は、CPU130に対して割込み状態信号138を生成する役割を有する。割込み状態信号138は、プロセッサ100に対して現在如何なる種類の割込みが発生しているかを示す信号である。通常、ステータスレジスタ134は現在の割込みマスクに関する情報を有しており、割込み状態信号138で指定される割込みを受け付けるか否かを決定する。割込みを受け付ける場合、割込みコントローラ135は、プログラムカウンタ133，ステータスレジスタ134などの値を書き替え、対応する割込み処理プログラムを実行させる。

【0034】

ステータスレジスタ134の構成を図3に示す。ここでは、プロセッサ100が全割込みマスク機能と割込みレベルマスク機能とを装備している場合を示している。図3では、ステータスレジスタ134が割込みブロックビット140と割込みマスクレベルフィールド141を有する。割込みブロックビット140がONである場合、プロセッサ100に対する全ての割込みがマスクされる。割込みマスクレベルフィールド141は現在の割込みマスクレベル値を示し、これ以下の割込みレベルは受け付けられない。図3では、割込みマスクレベルフィールド141は4ビット長である。このため、合計16種類のマスクレベルを指定可能であるが（通常、割込みレベル0は「割込みが発生していない」ことを意味し、割込みマスクレベルを0とすることは「割込みマスクを行わない」ことを意味するため、実質的には15種類となる）、割込みマスクレベルフィールド141のビット数を変更することにより、受理できる割込みレベルの種類を増減させることができる。

【0035】

図4は、プロセッサ100が全割込みマスク機能と個別割込みマスク機能とを

装備している場合のステータスレジスタ 134 の構成を示したものである。この例では、ステータスレジスタ 134 は実際には二つのレジスタ（実行状態レジスタ 142 と割込みマスクレジスタ 143）から構成される。図 3 と同様、実行状態レジスタ 142 内に割込みブロックビット 140 が設けられている。割込みマスクレジスタ 143 内の割込みマスクビット 144～147 はそれぞれ別々の割込みに対応しており、この割込みマスクビットのいずれかを ON とした場合、対応する割込みが受け付けられなくなる。図 3 に示すステータスレジスタは、図 4 のステータスレジスタの特殊なものである。例えば、割込みマスクビット 144 のみ ON となっている状態をレベル 1 とし、割込みマスクビット 144, 145 の二つが ON となっている状態をレベル 2、割込みマスクビット 144～146 の三つが ON となっている状態をレベル 3、…、のように対応させることができる。このため、以降、ステータスレジスタ 134 は図 4 に示す構成として、本発明の説明を行う。

【0036】

一般的なプロセッサでは、割込みを受理すると、ハードウェアによって自動的に割込みブロックビット 140 が ON に書き替えられ、プログラムカウンタ 133 に割込み処理プログラムのアドレスが格納される。必要に応じて、割込み処理プログラムが割込みブロックビット 140 を OFF に書き替えて割込み受け付けを許可することができる。また、オペレーティングシステムおよびタスクが、一時的に割込みブロックビット 140 や割込みマスクレジスタ 143 の内容を書き替え、特定割込みの受け付けを待たせることもできる。割込みマスク機能は、排他制御の実現や、割込み処理実行中に再び同一割込みが発生することを避けるために用いられる。

【0037】

次に、このようなハードウェア上に実現される本発明のソフトウェア構成について説明する。図 5 に、事務処理 OS 110, リアルタイム OS 111 の内部構成を示した。図中、全てのコンポーネントはメモリ 101 上に格納されている。事務処理 OS, リアルタイム OS とともに、実行可能タスクの情報を待ち行列 150, 151 の形で管理する。実行可能待ち行列 150, 151 は優先順位毎に設け

られる場合もあるが、本実施例では、全ての実行可能なタスクが一つの待ち行列によって管理される。なお、実行可能待ち行列が一つであっても、優先順位毎に設けられていても、本発明の内容には影響を与えない。各タスクは、(a) 実行状態、(b) 実行可能状態、(c) 待ち状態の三種の状態を順次とるのでオペレーティングシステムは、実行可能タスク以外に、待ち状態タスク、停止状態タスクなどを別の待ち行列で管理する。尚、図5ではこれらの待ち行列の記載を省略している。実行可能待ち行列150および151によって管理されるタスク管理テーブル160～165は、実行すべきタスクの優先順位、タスク実行時のプログラムカウンタ・ステータスレジスタ・汎用レジスタの値などを保存する。

【0038】

実行可能タスクを一つの待ち行列で管理する場合、実行可能待ち行列に登録されるタスク管理テーブルは優先順位の高い順に配置される。すなわち、次に実行すべきタスクの管理テーブルが待ち行列の先頭に来るように構成する。前述したように、本実施例で述べる事務処理OS110は、0～31までの優先順位を有し、優先順位の数値が大きいほど優先順位が高い。また、リアルタイムOS111は、0～255までの優先順位を有し、優先順位の数値が小さいほど優先順位が高い。このため、事務処理OSでは、タスク114（優先順位「31」）に対応する管理テーブル162が実行可能待ち行列150の先頭に配置され、以下、タスク113（優先順位「7」）の管理テーブル161、タスク112（優先順位「0」）の管理テーブル160の順に配置される。逆に、リアルタイムOSでは、タスク115（優先順位「0」）に対応する管理テーブル163が実行可能待ち行列151の先頭に配置され、次に、タスク116（優先順位「99」）の管理テーブル164、タスク117（優先順位「255」）の管理テーブル165の順に配置される。

【0039】

各オペレーティングシステムは、また、割込み処理を行う割込みハンドラ152、153、タスクに対してサービスを提供するシステムコールプログラム156、157、および、タスク切り替えを行うリスケジューラ154、155を有する。タスクの生成・削除・停止・再開や、外部割込み・内部割込み発生に伴ってタ

スク切り替えを行わなければならない場合、リスケジューラ 154, 155 が起動される。リスケジューラ 154, 155 は、直前に実行していたタスクの実行環境（レジスタなど）をタスク管理テーブルに格納した後に呼び出され、新たに実行すべきタスク決定して、その実行環境をタスク管理テーブルから取り出す。これをプログラムカウンタ、ステータスレジスタ、汎用レジスタなどに設定することにより、選択したタスクを実行する。

【0040】

リスケジューラ 154, 155 は、その内部に優先順位通知モジュール 120, 121 を有する。優先順位通知モジュール 120, 121 は、新たに実行すべきタスクの実行環境をレジスタに設定する直前に起動され、タスクの優先順位をオペレーティングシステム切り替えプログラム 118 内の優先順位変換モジュール 122, 123 に通知する。

【0041】

図 6 には、オペレーティングシステム切り替えプログラム 118 の内部構成を示したものである。優先順位変換モジュール 122, 123 は、各オペレーティングシステムでの優先順位を正規化優先順位に変更するため、優先順位変換テーブル 170, 171 をそれぞれ所有する。正規化優先順位は、本実施例では、0～255 の範囲の整数であり、「正規化優先順位の数値が大きいほど、正規化優先順位が高い」と定義する。勿論、正規化優先順位の数値の範囲を変更したり、「正規化優先順位の数値が小さいほど、正規化優先順位が高い」と定義することも可能である。

【0042】

前述したように、事務処理 OS 110 は 0～31 までの優先順位を有している。このとき、優先順位変換テーブル 170 は、32 個のエントリを有する配列となる。各配列要素は 0～255 の範囲の整数を有し、かつ、以下の不等式を満足する（ここでは、配列の名称を `prioBusiness` とする）。

【0043】

$$i > j \Leftrightarrow \text{prioBusiness}[i] > \text{prioBusiness}[j]$$

（ただし、 $0 \leq i, j \leq 31$ ）

事務処理 OS 110 の優先順位、および、正規化優先順位ともに、数値が大きいほど優先順位が高いとしているため、この条件が成立する必要がある。

【0044】

同様に、リアルタイム OS 111 が 0～255 までの優先順位を有している場合、優先順位変換テーブル 171 は、256 個のエントリを有する配列となる。各配列要素は 0～255 の範囲の整数を有し、かつ、以下の不等式を満足する（配列の名称を prioRealtime とする）。

【0045】

$$i > j \Leftrightarrow \text{prioRealtime}[i] < \text{prioRealtime}[j]$$

（ただし、 $0 \leq i, j \leq 255$ ）

事務処理 OS と不等号の向きが異なる理由は、リアルタイム OS では、優先順位の数値が小さいほど優先順位が高いとしているためである。

【0046】

図 6 では、事務処理 OS 110 上でタスク 114 が次に実行可能であると判定された場合、優先順位通知モジュール 120 から優先順位の数値「31」が通知される。優先順位変換モジュール 122 は、優先順位変換テーブル 170 を利用し、正規化優先順位の数値「124」を入手することができる。逆に、リアルタイム OS 111 上でタスク 115 が次に実行可能となった場合、優先順位通知モジュール 121 からリアルタイム OS での優先順位の数値「0」が通知される。優先順位変換モジュール 123 は、優先順位変換テーブル 171 を使用して、正規化優先順位の数値「255」を入手する。ここで、本実施例では、事務処理 OS 110 のタスクの優先順位がいくら高くても、正規化優先順位の数値が「124」を越えないことに注意する必要がある。これは、優先順位変換テーブルの設定方法の極端な例であり、リアルタイム OS 上のタスクをできる限り優先して実行させるといった目的のためのものである（リアルタイム OS 上のタスクでも、優先順位が低いものに対しては、事務処理 OS 上のタスクが優先することがある）。しかし、勿論、二つのオペレーティングシステムの優先順位ができるだけ対等に正規化優先順位に変換されるように、優先順位変換テーブル 170 の構成を変更することも可能である。

【0047】

優先順位変換モジュール122, 123が変換した正規化優先順位は、いずれも、優先順位比較モジュール124に通知される。優先順位比較モジュール124は、通知された事務処理OS正規化優先順位172とリアルタイムOS正規化優先順位173とを保持する。この場合、事務処理OS正規化優先順位172の数値は「124」であり、リアルタイムOS正規化優先順位173の数値は「255」である。

【0048】

前述したように、優先順位通知モジュール120, 121はリスケジューラ内に存在し、タスク切り替え時に実行される。タスク切り替えから次のタスク切り替えまでの期間は一つのタスクしか実行されないため、タスクの優先順位が動的に変化しない限り、この間、当該オペレーティングシステムに対する正規化優先順位も変化しない。優先順位通知モジュール120, 121は、タスク切り替え時に必ず優先順位を通知してくるため、優先順位比較モジュール124内に保持している事務処理OS正規化優先順位172とリアルタイムOS正規化優先順位173が、実際の各オペレーティングシステムでのタスクの優先順位を反映した数値となる。

【0049】

優先順位比較モジュール124は、保持している各オペレーティングシステムの正規化優先順位172, 173の数値を相互に比較して、より高い正規化優先順位を有するオペレーティングシステムを優先的に実行させる。図6の例では、より高い正規化優先順位を有するリアルタイムOS111を実行させることになる。

【0050】

オペレーティングシステム切り替えプログラム118は、優先順位変換モジュール122, 123, 優先順位比較モジュール124の他に、発生した割込みを各オペレーティングシステムに振り分ける共通割込みハンドラ174, オペレーティングシステム間での協調処理を実行するOS間通信機能モジュール175, 二つのオペレーティングシステムの実行環境切り替えを行うOSコンテキスト切

り替えモジュール 176, タスクの優先順位によって割込みマスクを変更する割込みマスクレベル計算モジュール 177 を有する。このプログラムの詳細は後述する。

【0051】

図 7 に事務処理 OS 110 におけるリスケジューラ 154 の処理フローを示した。リアルタイム OS 111 のリスケジューラ 155 も同様の処理を行う。リスケジューラ 154 は、一般に、(1) タイマ割込み処理終了時、(2) 外部割込み処理終了時、(3) システムコール実行時などにおいて、現在実行中のタスクが実行可能でなくなるか、実行中タスクより高い優先順位のタスクが実行可能となった場合、実行中タスクのレジスタをタスク管理テーブルに退避した後に起動されるモジュールである。リスケジューラ 154 を起動する可能性のあるシステムコールには、(a) タスクの生成・終了・停止・再開（例えば、自分より優先順位の高いタスクを生成した場合）、(b) 排他制御の実行・終了（排他制御待ち状態に移行した場合など）に加え、(c) タスクの優先順位変更（自己の優先順位を低下させた場合など）、といったものがある。

【0052】

リスケジューラ 154 は、第一に、実行可能待ち行列 150 から最高優先順位のタスク管理テーブルを取り出す（処理 181）。ここで、全てのタスクが待ち状態や停止状態となっていて、実行可能タスクが存在しないこともある。これを処理 182 で判定する。

【0053】

実行可能タスクが存在しなければ、優先順位変換モジュール 122 に idle 状態であることを通知する（処理 184）。このとき、実行可能タスクが存在しないのであるから、idle ループへ移行する（処理 186）。idle ループとは、実行すべき処理が現れるまで何も行わないプログラムである。ここで注意すべきことは、処理 184 において、優先順位変換モジュール 122 に対して idle 状態を通知したことである。事務処理 OS 110 が idle 状態であれば、優先順位比較モジュール 124 はリアルタイム OS 111 を優先して実行させる。したがって、両オペレーティングシステムが同時に idle 状態に移行した場合を除いて、実際に処理

186でidleループが実行されることはない。再び事務処理OS110が実行される場合には、何らかの実行すべき処理が存在するのであるから、直ちにidleループを抜けて処理181に移行することになる。

【0054】

処理182で実行すべきタスクが存在すると判定した場合、次に実行すべきタスクの優先順位をタスク管理テーブルから取り出して、優先順位変換モジュール122に通知する。通知した優先順位がリアルタイムOS111の優先順位よりも低かった場合（正規化優先順位による比較）、この時点で、リアルタイムOS111に切り替えられることになる。リアルタイムOS111から逆に事務処理OS110へ切り替えが行われる場合、割込み処理などによって、実行しようとしているタスクが実行可能でなくなったり、更に高い優先順位のタスクが実行可能となったりしない限り、処理183の直後から再実行することになる（処理181で選択したタスクを起動する）。なお、当然のことながら、現在実行中のタスクが実行可能でなくなるか、実行中タスクより高い優先順位のタスクが実行可能となった場合には、リスケジューラ154自体が再起動される。処理185で、タスク管理テーブルからレジスタを復帰してプロセッサ100のレジスタに登録することにより、選択したタスクの処理を実行する。

【0055】

ここで、リスケジューラ154中、処理183、184が優先順位通知モジュール120に相当する。リスケジューラ155内にも同様に優先順位通知モジュール121が埋め込まれる。

【0056】

次に、図8に、事務処理OS110に対応する優先順位変換モジュール122の処理フローを示す。優先順位変換モジュール122は優先順位通知モジュール120から呼び出され、直ちに、優先順位変換処理を実行する。最初に、優先順位通知モジュール120がidle状態を通知してきたか否かを判定する（処理190）。idle状態はあらゆる優先順位よりも低い優先順位を有すると見なすこともできるので、ここでは、いずれの正規化優先順位よりも低くなるように、（-1）という正規化優先順位の数値を有していると考え。処理193では、この数値

(-1) を優先順位比較モジュール 124 に通知する。idle 状態でなければ、優先順位変換テーブル 170 から対応するエントリを読み出し (処理 191)、これを優先順位比較モジュール 124 に通知する (処理 192)。リアルタイム OS 111 に対応する優先順位変換モジュール 123 の処理も同様である。

【0057】

図 9 は優先順位比較モジュール 124 の処理フローである。優先順位比較モジュール 124 は、第一に、事務処理 OS 110 からの正規化優先順位であるか、リアルタイム OS 111 からの正規化優先順位であるかを判定する (処理 200)。事務処理 OS 110 の正規化優先順位であれば、事務処理 OS 正規化優先順位 172 に入手した正規化優先順位を格納する (処理 201)。リアルタイム OS 111 からの正規化優先順位であれば、リアルタイム OS 正規化優先順位 173 に当該正規化優先順位を記憶する (処理 202)。

【0058】

正規化優先順位が通知されてくるのは、いずれか一方の正規化優先順位が変化した可能性がある場合のみであるため、ここで、事務処理 OS 正規化優先順位 172 とリアルタイム OS 正規化優先順位 173 の数値を比較する (処理 203)。本実施例では、正規化優先順位の数値が大きいオペレーティングシステムを優先的に起動するとしている。したがって、事務処理 OS 110 の正規化優先順位の数値がリアルタイム OS 111 の正規化優先順位の数値より大きければ、次に実行すべきオペレーティングシステムを事務処理 OS 110 とする (処理 204)。これ以外の場合には、次にリアルタイム OS 111 を実行させるものとする (処理 205)。なお、ここでは、事務処理 OS 110 とリアルタイム OS 111 の正規化優先順位が等しくても、リアルタイム OS 111 を実行させる。これを変更し、正規化優先順位が互いに等しい場合には事務処理 OS 110 を実行させるとしても問題はない。また、優先順位変換テーブル 170、171 の内容を設定するにあたって、各オペレーティングシステムから得られる正規化優先順位が互いに等しくならないようにすることも、システムを単純化させるための一つの方法である。

【0059】

次に処理206で、新たに実行すべきオペレーティングシステムが今まで実行していたオペレーティングシステムに等しいか否かを判定する。今までと異なったオペレーティングシステムを実行させる場合、OSコンテキスト切り替えモジュール176に依頼して、オペレーティングシステム実行環境の退避・復帰を行わせる（処理207）。

【0060】

図10に割込みハンドラ152, 153, 共通割込みハンドラ174, OS間通信機能モジュール175, OSコンテキスト切り替えモジュール176の詳細を示す（割込みマスクレベル計算モジュール177については、別の図面で詳細を説明する）。

【0061】

割込みハンドラ152, 153は、割込み発生時のレジスタなどの保存、一時変数の保持を行うための領域として、それぞれ、割込みスタック214, 215を所有する。割込み発生時には、割込み発生直前のレジスタ値を保存し、このレジスタ値を割込み処理終了後に復帰してやらなければならない。このため、割込みスタック214, 215が必要とされる。使用するプロセッサ100の種類によっては、割込み発生時に自動的にレジスタが切り替わり、かつ、割込み処理終了後に切り替え前のレジスタに戻す機能を備えているものもある。しかしながら、多重割込みが可能なシステムを考慮した場合には、このようなハードウェアを使用していても、割込みスタックが必要となる（割込み処理中により高い緊急性を有する割込みが発生した場合、新たに発生した割込み処理を優先的に実行し、次いで、元の割込み処理に戻る必要がある）。

【0062】

割込みハンドラ152, 153は、更に、割込みスタック214, 215中、どの領域まで使用したかを示すためのポインタとして、割込みスタックポインタ216, 217を所有する。割込みハンドラ152, 153は、割込みスタック、および、割込みスタックポインタを用いて割込み発生前の実行環境（レジスタなど）を保存し、必要な割込み処理を実行する。割込み処理終了後、割込み発生

前の実行環境を復帰して元々動作していたプログラムの実行を継続させる。

【0063】

共通割込みハンドラ 174 は、発生した割込みを割込みハンドラ 152, 153 に対して振り分ける役割を有する。このため、共通割込みハンドラ 174 は、現在実行しているオペレーティングシステムが事務処理 OS 110 であるかリアルタイム OS 111 であるかを記憶する領域として、実行 OS 記憶変数 210 を所有する。例えば、図 10 に示した時点で事務処理 OS が処理を実行しているものと仮定すると、この場合、実行 OS 記憶変数 210 には、「事務処理 OS」が記憶されている。勿論、実行 OS 記憶変数 210 が文字列「事務処理 OS」を記憶することは非常に非効率的であるため、例えば、

- ・事務処理 OS → 0
- ・リアルタイム OS → 1

といった整数形式で記憶するとよい。共通割込みハンドラ 174 は、更に、発生した割込みがいずれのオペレーティングシステムに対応するかを示す割込み対応テーブル 211 を有する。割込み対応テーブル 211 は、個々の割込みがどちらのオペレーティングシステムで処理されるべきかを示す対応表である。本実施例では、図 4 に示したように、32 種類の割込み要因が存在する。このため、割込み対応テーブル 211 も 32 個のエントリ (0 ~ 31) から構成される。図 10 の例では、割込み要因 0 が事務処理 OS で、割込み要因 1 がリアルタイム OS で、…、割込み要因 31 がリアルタイム OS でというように、いずれのオペレーティングシステムで処理が実行されるべきかが定義されている。なお、効率上、割込み対応テーブル 211 の内容も、実行 OS 記憶変数 210 と同様、文字列形式ではなく、整数形式で記憶すべきである。

【0064】

さて、計算機の種類によっては、一つの入出力機器を二つのオペレーティングシステムが共用することもある（例えば、事務処理 OS 110, リアルタイム OS 111 の両者がディスプレイに文字や映像を出力する場合）。このようなシステムを実現するためには、割込み振り分け先のオペレーティングシステムを動的に変更できることが必要となる。本実施例で、振り分け先オペレーティングシ

システムを固定化するのではなく、割込み対応テーブル 211 から入手しているのは、このためである。割込み対応テーブル 211 の内容を入出力機器の使用状況によって変更することにより、割込みの振り分け先を動的に変更することができる。

【0065】

共通割込みハンドラ 174 は、実行 OS 記憶変数 210 を用いて現在実行しているオペレーティングシステムの種類を判定し、これが割込み対応テーブル 211 から得られた振り分け先オペレーティングシステムに一致しなければ、OS コンテキスト切り替えモジュール 176 に切り替えを依頼する。オペレーティングシステムが一致するか、オペレーティングシステム切り替え終了後、該当するオペレーティングシステムの割込みハンドラに処理を依頼する。なお、実行 OS 記憶変数 210 は OS コンテキスト切り替えモジュール 176 から参照される。このように、オペレーティングシステム切り替えプログラム 118 内の各モジュールが有する内部構造は、個々のモジュールによって占有されるものとは限らず、各モジュールが必要に応じて共用できる。

【0066】

OS コンテキスト切り替えモジュール 176 は、優先順位比較の結果、若しくは、割込み発生の結果、オペレーティングシステムを切り替えなければならない場合に起動される。二つのオペレーティングシステムを切り替えるため、これらの実行環境（すなわち、レジスタ値）を保存しておくための領域を所有しなければならない。ここでは、事務処理 OS の実行環境を保存するための領域として、事務処理 OS 用保存コンテキスト 212、リアルタイム OS の実行環境を保存するための領域として、リアルタイム OS 用保存コンテキスト 213 を用意する。OS コンテキスト切り替えモジュール 176 は、現在実行しているオペレーティングシステムに対応する保存コンテキストに実行環境を保存し、次に、もう一方の保存コンテキストから実行環境を読み出して、プロセッサ 100 のレジスタに設定する。これにより、オペレーティングシステムの切り替えを実施できる。

【0067】

OS 間通信機能モジュール 175 は、二つのオペレーティングシステム上のタ

スクが相互に通信・協調するためのプログラムである。ここでは、事務処理 OS 110, リアルタイム OS 111 の両者が使用できる共有メモリ 218、並びに、オペレーティングシステム間で排他制御を行うためのロック取得モジュール 219, ロック解放モジュール 220 が存在するとしている。共有メモリ 218 は、メモリ 101 の一部であり、両オペレーティングシステムから参照を行える領域である。共有メモリ 218 以外のメモリは、基本的に、事務処理 OS 用領域, リアルタイム OS 用領域, オペレーティングシステム切り替えプログラム用領域に分割され、それぞれ、事務処理 OS 110, リアルタイム OS 111, オペレーティングシステム切り替えプログラム 118 に占有されて使用される。二つのオペレーティングシステム上のタスクが共有メモリ 218 を使用する場合、通常、排他制御を行って共有メモリ 218 上のデータの一貫性を保つ。アプリケーションプログラムがオペレーティングシステム間に跨る排他制御を行う場合、ロック取得モジュール 219, ロック解放モジュール 220 が提供する機能を利用する。

【0068】

このとき、優先順位逆転現象と呼ばれる状況について注意しておかなければならない。優先順位逆転現象とは、一般に、以下の状態を意味する。

【0069】

- (1) 低優先順位であるタスク α が最初に起動されてロックを取得する。

【0070】

- (2) 次に高優先順位であるタスク β が起動されてロック待ち状態に移行する。
- (3) 次に中優先順位であるタスク γ が起動され、タスク α (低優先順位) からタスク γ (中優先順位) への切り替えが行われる。

【0071】

このとき、タスク γ の実行によってタスク α の実行が妨げられ、結果的に、高優先順位タスク β がロックを取得するまでの時間が延びてしまうことになる。優先順位逆転現象は、通常、優先順位上昇方式または優先順位継承方式を用いて解決する。

【0072】

優先順位上昇方式は、ロック取得を行ったタスクの優先順位を一定のレベルまで高くする方式である。これによって、ロック取得を行ったタスク α の優先順位が一時的にタスク γ （中優先順位）よりも高くなり、タスク γ が起動されても、ロック解放まではタスク α が実行される。この後、高優先順位であるタスク β がロックを取得し、処理を継続することになる。

【0073】

優先順位継承方式は、前記優先順位上昇方式を変更して柔軟性を持たせたものである。優先順位継承方式では、ロック待ち状態に移行したタスク（例では、タスク β ）の優先順位がロックを取得しているタスク（タスク α ）よりも高ければ、この優先順位をロック取得中タスクに継承させる方式である。ここでは、ロック取得中の間だけ、タスク α がタスク β の優先順位を受け継ぐことになる。低優先順位のタスク同志で排他制御を行う場合など、必ずしも優先順位を一定レベルまで上昇させる必要がない場合もあり、優先順位継承方式は、このような状況にも対処できる方式である。

【0074】

本実施例で述べるロック取得モジュール219，ロック解放モジュール220は、このような優先順位上昇もしくは優先順位継承方式をオペレーティングシステム間で実施できるものとする。オペレーティングシステム間での優先順位上昇・継承は、正規化優先順位を用いて行われる。

【0075】

以降、図10で示したモジュールのうち、OSコンテキスト切り替えモジュール176，共通割込みハンドラ174，割込みハンドラ152，ロック取得モジュール219，ロック解放モジュール220の処理フローの説明を行う。割込みハンドラ153に関しては、割込みハンドラ152と同様の処理フローを有するため、説明を省略する。

【0076】

図11はOSコンテキスト切り替えモジュール176の処理フローである。

【0077】

OS コンテキスト切り替えモジュール 176 は、オペレーティングシステムを切り替えなければならない場合にのみ呼び出され、切り替えが必要か否かのチェックは呼出し前に行われている。したがって、最初に、どちらのオペレーティングシステムへ切り替えなければならないかのみをチェックする（処理 230）。ここで、事務処理 OS 110 からリアルタイム OS 111 への切り替えを行う場合（現実行オペレーティングシステムが事務処理 OS 110 であって、リアルタイム OS 111 の優先順位が高くなった場合）、事務処理 OS 用保存コンテキスト 212 内に使用中レジスタの値を退避する（処理 231）。次に、リアルタイム OS 用保存コンテキスト 213 から実行環境を復帰して、レジスタに設定することにより（処理 232）、以前退避した状態からリアルタイム OS 111 を再実行させることができる。リアルタイム OS 111 から事務処理 OS 110 への切り替えを行う場合、逆に、リアルタイム OS 用保存コンテキスト 213 に使用中レジスタの値を退避し（処理 233）、次に、事務処理 OS 用保存コンテキスト 212 からレジスタを復帰する（処理 234）。いずれの場合においても、最後に、切り替え後のオペレーティングシステムの種類を実行 OS 記憶変数 210 に書き込む（処理 235）。

【0078】

図 12 には、共通割込みハンドラ 174 の処理フローを示している。一般に、単一のオペレーティングシステムによって制御される計算機では、一旦、全ての割込みを割込みハンドラと呼ばれるモジュールが処理し、次いで、各プログラムに振り分ける。しかしながら、本実施例で説明するような複数のオペレーティングシステムを動作させる計算機では、この更に前に、共通割込みハンドラが全ての割込みを受け付け、これに対応するオペレーティングシステムの割込みハンドラに振り分ける形となる。各オペレーティングシステムに割込みを振り分けるに当たって、割込み発生時に、割込み対象とは別のオペレーティングシステムが処理を実行していることもある。この場合、割込みに対応するオペレーティングシステムに切り替える必要があり、共通割込みハンドラはこのような役割も有することになる。

【0079】

共通割込みハンドラ 174 は、割込み発生後、第一に、実行 OS 記憶変数 210 の内容を取り出して、現在実行中のオペレーティングシステムが事務処理 OS 110 であるかリアルタイム OS 111 であるかをチェックする（処理 240）。次に、割込み対応テーブル 211 を用いて、発生した割込みがいずれのオペレーティングシステムに対応しているかを入手する（処理 241）。例えば、図 10 の割込み対応テーブル 211 を用いるとすると、割込み「0」が発生した場合には事務処理 OS 110、割込み「1」が発生した場合にはリアルタイム OS 111、…、割込み「31」が発生した場合にはリアルタイム OS 111、というように対応するオペレーティングシステムを入手することができる。ここで、割込み対象オペレーティングシステムが実行中オペレーティングシステムに等しいか否かをチェックする（処理 242）。

【0080】

発生した割込みが現在実行中のオペレーティングシステムに対するものでなければ、一旦、オペレーティングシステムの切り替えを行わなければならない。この切り替え処理は、OS コンテキスト切り替えモジュール 176 に依頼することによって行う（処理 243）。次に割込み対象オペレーティングシステムが事務処理 OS 110 であるかリアルタイム OS 111 であるかをチェックし（処理 244）、対象が事務処理 OS 110 であれば、事務処理 OS の割込みハンドラ 152 を起動する（処理 245）。リアルタイム OS 111 への割込みが発生していれば、リアルタイム OS の割込みハンドラ 153 を起動することになる（処理 246）。一般に、割込みハンドラ 152、153 は、割込み処理によってタスク切り替えを行わなければならないとき、リスケジューラ 154、155 を呼び出し、共通割込みハンドラ 174 に制御を戻すことはない。しかし、タスク切り替えが発生しない場合、そのまま割込みハンドラの処理を終了する。このとき、割込みハンドラ 152、153 は、共通割込みハンドラへ制御を戻し（図 13 で後述）、共通割込みハンドラは処理 247 から動作を再開する。処理 247 は、割込み発生時にオペレーティングシステムを切り替えたかどうかをチェックする処理である。処理 243 によってオペレーティングシステムを切り替えた場合

、ここで再び、オペレーティングシステムを切り替え、元の実行環境に戻す必要がある。タスク切り替えが発生しないということは、両オペレーティングシステムの正規化優先順位が変化しなかったということを意味し、割込み発生前の実行状態に戻さなければならないわけである。このため、もう一度OSコンテキスト切り替えモジュール176に依頼して、切り替え処理を実行させる(処理248)

【0081】

各オペレーティングシステムの割込みハンドラがリスケジューラ154, 155を実行した場合、前述したように、共通割込みハンドラの処理247に制御が戻ることはない。この場合、リスケジューラ154, 155が新たなタスクの優先順位を優先順位変換モジュール122, 123に通知してくる。これによって、優先順位比較モジュール124がオペレーティングシステムの切り替えを行うか否かの判定を行う。

【0082】

なお、共通割込みハンドラ174の構成を図28のように変更することも可能である。図29に示した共通割込みハンドラ174は、図10に示した共通割込みハンドラの構成要素の他に、割込み優先順位対応テーブル380を所有する。割込み優先順位対応テーブル380は、割込みハンドラがいかなる正規化優先順位で動作しなければならないかをしめす対応表である。図29の例では、割込み要因0に対応する割込みハンドラは、正規化優先順位255で動作しなければならない。また、割込み要因31に対応する割込みハンドラは、正規化優先順位224で動作しなければならない。共通割込みハンドラ174は各オペレーティングシステムの割込みハンドラを起動する際に、割込み優先順位対応テーブル380にしたがって、当該オペレーティングシステムの正規化優先順位を更新し、また、割込みハンドラ終了後、元の正規化優先順位を復帰させる。なお、この例では共通割込みハンドラ174が割込み優先順位対応テーブル380を有するものとしているが、割込み優先順位対応テーブル380を優先順位変換モジュール122, 123内に設け、共通割込みハンドラ174から優先順位変換を依頼する構成を採ることも可能である。

【0083】

なお、図29のように、割込みに対しても正規化優先順位を割当てることができ、各オペレーティングシステムのあらゆる動作状態にも正規化優先順位を割当てることができる。例えば、オペレーティングシステムは、一般に以下の四種類の動作状態を持つ。

【0084】

- (1) idle状態
- (2) タスク実行状態
- (3) オペレーティングシステム自体の処理状態（例えば初期化処理）
- (4) 割込み処理状態

これら全てに正規化優先順位を割当てることが可能である。この場合、一般に考えて、割込み処理状態が最も優先して動作しなければならない、最も高い正規化優先順位を割当て、次にオペレーティングシステム自体の処理状態、タスク実行状態、idle状態の順に正規化優先順位を割当てていくことになる。

【0085】

図13は共通割込みハンドラ174から呼び出される事務処理OSの割込みハンドラ152の処理フローを示したものである。第一に、使用中レジスタを割込みスタック214に退避し（処理250）、割込み処理を実行する（処理251）。ここで、オペレーティングシステムがリスケジューリング中か否か（すなわち、リスケジューラ154の処理実行中か否か）をチェックする（処理252）。リスケジューリング中とは、次に実行すべきタスクを選択している最中であり、実際にはタスクを実行していないことを意味する。したがって、オペレーティングシステムの処理を単純化させるならば、この場合、再びリスケジューリングを最初からやり直せば良いことになる。すなわち、リスケジューリング中であると判定した場合、割込みスタックの退避レジスタを破棄し（処理257）、リスケジューラ154を最初から起動する（処理258）。なお、リスケジューリング中に割込みが発生しても、実行すべきタスクを新たに選択し直す必要がない場合もある。本方式は、この状況でもリスケジューリングを最初から行うことになり、効率的でないこともある。このため、リスケジューリング中に発生し、リスケ

ジューラ 154 の実行に影響を与える処理（例えば、タスク起動・終了など）を待ち行列などに登録しておく方法を用いることもできる。この場合、リスケジューラ 154 の終了前や優先順位通知モジュール 120 内で、該待ち行列に登録しておいた処理を纏めて実行する。この方式を採用すると、割込み発生たびに途中まで実行していたリスケジューリングをやり直す必要が無い。しかしながら、当該待ち行列に登録していた処理を纏めて実行した後、これによってリスケジューリングが再び必要となるか否かを再チェックしなければならない。

【0086】

なお、本実施例では、単純化のため、前者の方式で説明を行う。しかし、後者の方式を用いる場合、

(a) リスケジューリング中か否かを示すフラグ

(b) 処理待ち行列

を用意すればよい。オペレーティングシステムが提供するシステムコールなどにおいて、リスケジューラ 154 の実行に影響を与える処理を、リスケジューリング中であれば、該待ち行列に登録する。更に、リスケジューラ 154 の処理フロー中に、該待ち行列に登録された処理を一括して実行するモジュールを挿入する。

【0087】

割込みハンドラ 152 がリスケジューリング中でないと判定した場合、当該オペレーティングシステムはその時点で何らかのタスクを実行している。このため、第一に、タスク切り替えが必要か否かを判定する（処理 253）。タスク切り替えが必要なければ（現実行タスクが最高優先順位であり、かつ、実行可能であれば）、このまま、割込みハンドラ 152 の処理を終了する。このため、割込みスタック 214 上に退避しておいたレジスタ値を復帰し（処理 254）、共通割込みハンドラへ制御を戻す（処理 255）。タスク切り替えが必要であると判定した場合、割込みスタック 214 上に退避したレジスタの値をタスク管理テーブルにコピーし（処理 256）、割込みスタック上の退避レジスタを破棄する（処理 257）。この後、リスケジューラ 154 を起動する（処理 258）。なお、プロセッサ 100 に、メモリ 101 上データの参照を行って、同時に、割込みス

タックポインタ 216 の値を増減できる機能が備わっていれば、処理 256 と処理 257 を纏めて実行することができる。

【0088】

次に、ロック取得モジュール 219、ロック解放モジュール 220 の処理フローを説明する。ここでは、優先順位上昇方式を用いる例を説明し、優先順位継承方式を使用する実施例については後述する。なお、ここでは、ロック取得・解放を要求したタスク、および、それを管理するオペレーティングシステムを自タスク、自オペレーティングシステムと呼び、もう一方のオペレーティングシステムを別オペレーティングシステムと呼んでいる。

【0089】

図 14 が優先順位上昇方式を用いるロック取得モジュール 219 の処理フローである。ロック取得モジュール 219 は、第一に、別オペレーティングシステムのタスクがロック取得中か否かをチェックし（処理 260）、取得中でなければ、自オペレーティングシステムがロックの所有者であるように設定する（処理 261）。別オペレーティングシステムのタスクがロック取得中であれば、ロックが解放されるまでタスクを待ち状態に移行させる（処理 263）。このとき、別オペレーティングシステムを優先して実行させ、当該ロックを速やかに解放させるようにしなければならない。このため、優先順位比較モジュール 124 内に記憶されている別オペレーティングシステムの正規化優先順位を自オペレーティングシステムより高くなるように変更する（処理 262）。ここで処理 263 においてタスクを待ち状態に移行させると、自オペレーティングシステムでリスクジェーラが起動される。これにより、リスクジェーラから、優先順位通知モジュール、優先順位変換モジュール、優先順位比較モジュールと順次呼び出され、正規化優先順位を高く変更した別オペレーティングシステムに切り替えが行われることになる。

【0090】

図 15 が優先順位上昇方式を用いる場合のロック解放モジュール 220 の処理フローである。第一に、自己が所有するロックを解放する（処理 270）。次に、別オペレーティングシステムがロック取得待ちか否かをチェックする（処理

271)。ロック取得待ちでなければ、正規化優先順位の上昇は行われていない。このため、このままロック解放モジュール220の処理を終了する。しかし、別オペレーティングシステムがロック取得待ちであれば、優先順位逆転現象を解決するため、正規化優先順位の上昇が行われている筈である。したがって、自オペレーティングシステムの正規化優先順位を元の値に戻し（処理272）、次に、別オペレーティングシステムでロック取得待ちとなっているタスクを実行可能状態に移行させる（処理273）。ロック取得待ちタスクを実行可能とすることによりタスク切り替えが発生すれば、別オペレーティングシステムのリスケジューラが起動され、前述のロック取得モジュールの場合と同様、最終的に、優先順位比較モジュール124が実行される。これにより、元の正規化優先順位を用いてオペレーティングシステムの重要性が比較され、より高い正規化優先順位を有するオペレーティングシステムが選択される。

【0091】

次に、優先順位継承方式を用いる例を説明する。この場合、図16に示すように、各オペレーティングシステムは、正規化優先順位にしたがってタスクの優先順位を変更する優先順位設定モジュール280、281を装備しなければならない。また、優先順位変換モジュール122、123は、優先順位を正規化優先順位に変換する機能のほか、正規化優先順位を優先順位に変換する機能を搭載しなければならない。このため、優先順位変換モジュール122、123は、正規化優先順位をインデクスとし、各オペレーティングシステムの優先順位を内容とする優先順位逆変換テーブル282、283を有する。

【0092】

オペレーティングシステムは、一般に、システムコールとして、タスクの優先順位を変更する機能を有している。優先順位設定モジュール280、281は、与えられた正規化優先順位を各オペレーティングシステムに対応する優先順位に変換し、タスク優先順位変更システムコールなどを用いてタスクの優先順位を変更する。優先順位逆変換テーブル282、283は、優先順位設定モジュール280、281の機能をサポートする役割を有する。本実施例では、正規化優先順位が0～255であるため、優先順位逆変換テーブル282、283は、256

個のエントリを有する配列である。また、事務処理OS 110の優先順位の範囲が0～31であるため、優先順位逆変換テーブル282の各配列要素は、0～31の範囲のいずれかの整数を有し、かつ、以下の不等式を満足する（配列の名称をrevBusinessとする）。

【0093】

$$i > j \Rightarrow \text{revBusiness}[i] \geq \text{revBusiness}[j]$$

（ただし、 $0 \leq i, j \leq 255$ ）

リアルタイムOS 111は優先順位0～255を有し、かつ、優先順位の数値が小さいほど優先順位が高いという特徴を有するため、優先順位逆変換テーブル283の各配列要素は、0～255までのいずれかの整数を有し、かつ、以下の不等式を満足する（配列の名称をrevRealtimeとする）。

【0094】

$$i > j \Rightarrow \text{revRealtime}[i] \leq \text{revRealtime}[j]$$

（ただし、 $0 \leq i, j \leq 255$ ）

さて、本実施例では、正規化優先順位の数値が0～255、リアルタイムOSの優先順位の数値が0～255であるとしているため、正規化優先順位とリアルタイムOSの優先順位との対応付けを一对一に行うことができている。したがって、優先順位変換テーブル171から一意に優先順位逆変換テーブル283を導出することができる。しかしながら、事務処理OSの優先順位の数値は0～31であるため、優先順位逆変換テーブル282を一意に定めることはできない。この場合、例えば、正規化優先順位「1」に対応する事務処理OSの優先順位を、優先順位変換テーブル170から導き出すことができない。このため、一意に定めることのできない正規化優先順位に対しては、上記不等式を満たすように、事務処理OSの優先順位を適宜に定めている。

【0095】

優先順位変換モジュール122のうち、優先順位逆変換機能の処理フローについて図17に示した。ここでは、指定正規化優先順位に対応する事務処理OSの優先順位を優先順位逆変換テーブル282から取り出し（処理290）、当該優先順位を優先順位設定モジュール280に通知する（処理291）。優先順位変

換モジュール 123 も同様の優先順位逆変換機能を有する。

【0096】

これに対する優先順位設定モジュール 280 の処理フローを図 18 に示す。優先順位設定モジュール 280 は、第一に、優先順位変換モジュール 122 に依頼して、正規化優先順位から事務処理 OS の優先順位への変換を行わせる（処理 292）。次に、事務処理 OS の優先順位変更システムコールなどを用いてタスクの優先順位を変更させる（処理 293）。リアルタイム OS 111 における優先順位設定モジュール 281 の処理も同様である。

【0097】

次に、これらの優先順位逆変換機能、優先順位設定モジュールを用いた優先順位継承方式の実現方法について述べる。

【0098】

図 19 は優先順位継承方式を用いるロック取得モジュール 219 の処理フローである。ロック取得モジュール 219 は、第一に、別タスクがロック取得中か否かをチェックし（処理 300）、取得中でなければ、自タスクがロックの所有者であるように設定する（処理 301）。別タスクがロック取得中であれば、ロックが解放されるまでタスクを待ち状態に移行させる必要がある（処理 304）。このとき、ロック取得中タスクを優先して実行させ、当該ロックを速やかに解放させるようにしなければならない。ロック取得中タスクの正規化優先順位と自タスクの正規化優先順位を比較し（処理 302）、ロック取得中タスクの正規化優先順位が低ければ、当該タスクに自タスクの優先順位を継承させる（処理 303）。優先順位継承は、ロック取得中タスクを実行するオペレーティングシステムの優先順位設定モジュールに対し、自タスクの正規化優先順位を指定して優先順位設定を依頼することによって行うことができる。ロック取得中タスクの正規化優先順位が既に自タスクの正規化優先順位より高ければ、優先順位継承を行う必要はなく、直ちにタスクを待ち状態に移行させることができる。

【0099】

図 20 は優先順位継承方式を用いるロック解放モジュール 220 の処理フローである。第一に、自己が所有するロックを解放する（処理 310）。次に、別タ

スクがロック取得待ちか否かをチェックする（処理 311）。ロック取得待ちでなければ、正規化優先順位継承が行われていないことになり、このままロック解放モジュール 220 の処理を終了する。しかし、別タスクがロック取得待ちであれば、正規化優先順位継承が行われている筈である。したがって、優先順位設定モジュールに依頼して自タスクの正規化優先順位を元の値に戻し（処理 312）、次に、ロック取得待ちとなっているタスクを実行可能状態に移行させる（処理 313）。

【0100】

本実施例では説明を省略したが、ロック取得モジュール 219、ロック解放モジュール 220 などを実行している途中に割込みが発生し、タスク切り替えなどが起こると、ロック所有者を示すデータなどに不整合が生じる可能性がある。したがって、通常、これらの処理実行中には、割込みマスク機能によって割込み発生を抑止する。

【0101】

図 16～図 20 に示したプログラムモジュール、処理フローにより、オペレーティングシステム間での排他制御において、優先順位継承方式を用いることが可能となる。なお、実際には、図 19、図 20 に示した処理フローを用いることにより、一つのオペレーティングシステム内で正規化優先順位の継承を行うことも可能である。

【0102】

図 21 に、割込みマスキングレベル計算モジュール 177 の内部構造を示した。割込みマスキングレベル計算モジュール 177 は、正規化優先順位を割込みマスキングレベルに変換し、一定以上の正規化優先順位であれば、特定の割込みをマスクする機能である。これは、ある一定以上の正規化優先順位を有するタスクは、割込み処理よりも優先して実行されなければならないという考えに基づく機能である。したがって、システムコールなどで割込みをマスクする機能が備わっていれば、本機能が存在しなくても十分システムは成立する。ただし、割込みマスキングレベル計算モジュール 177 の利点として、正規化優先順位が一定値以上になれば、自動的に割込みマスクを実行することがある。

【0 1 0 3】

割込みマスクを実行するため、割込みマスクレベル計算モジュール 1 7 7 は、割込みマスク変換テーブル 3 2 0 を所有する。割込みマスク変換テーブル 3 2 0 は、正規化優先順位から割込みマスク値を入手するための変換テーブルである。本実施例では、図 4 に示したように、3 2 種類の割込み要因を個別にマスクできる計算機アーキテクチャを対象としているため、割込みマスク変換テーブル 320 の各エントリは 3 2 ビット長データを有する。しかしながら、図 3 に示したように、4 ビットの割込みマスクレベルを使用すると、割込みマスク変換テーブル 3 2 0 の各エントリを 4 ビット長に縮小させることができる。ここで、図 2 1 に示した実施例では、例えば、正規化優先順位「0」に対応する割込みマスク値が 0x00000000 であり、正規化優先順位「2 5 5」に対応する割込みマスク値は 0xffffffff である。これは、正規化優先順位が「0」であれば全ての割込みを許可し、最高の正規化優先順位でシステムが動作しているならば、全ての割込みをマスクすることを意味する。優先順位比較モジュール 1 2 4 が割込みマスクレベル計算モジュール 1 7 7 に対して、現在実行中オペレーティングシステムの正規化優先順位を通知することにより、必要な割込みマスクを自動的に行うことができる。

【0 1 0 4】

図 2 2 に割込みマスクレベル計算モジュール 1 7 7 の処理フローを示す。ここでは、第一に、正規化優先順位に対応する割込みマスク値を割込みマスク変換テーブル 3 2 0 から取り出す（処理 3 3 0）。次に、入手した割込みマスク値を割込みマスクレジスタ 1 4 3 などに設定して、割込みマスクを実施する（処理 331）。

【0 1 0 5】

以上に本発明の第一の実施例について説明した。本実施例を用いることにより、各オペレーティングシステムが実行しているタスクの優先順位にしたがって、オペレーティングシステムの切り替えを行うことができる。本実施例では、各オペレーティングシステム内に優先順位通知モジュールを組み込み、リスケジューリングのたびに優先順位の変更を通知するように改造しなければならない。しか

しながら、市販オペレーティングシステムを使用する場合など、このような改造を施せないことも多い。以下、これに対処する本発明の第二の実施例について説明する。

【0106】

図23は本発明の第二の実施例を示した構成図である。事務処理OS110、リアルタイムOS111は、それぞれ、現在実行中タスクの優先順位（実行優先順位340、341）を保持するものとする。各オペレーティングシステムは、図5に示す実行可能待ち行列150、151の先頭に存在するタスク管理テーブル162、163内に格納されている優先順位の数値が実行優先順位340、341となる。また、実行可能待ち行列が優先順位毎に複数個存在する場合には、一般に、現時点で最高優先順位の実行可能待ち行列がどこに存在するかを示すポインタや、最高優先順位の数値そのものを示す変数（優先順位保持変数）を用意することが多い。前者の場合には、該ポインタから実行可能待ち行列を辿り、タスク管理テーブルを参照することにより、後者の場合には、該優先順位保持変数自体を読み出すことにより、実行優先順位340、341を入手できる。

【0107】

オペレーティングシステム切り替えプログラム118内には、新たに、優先順位監視モジュール344を設ける。これ以外のモジュールである、優先順位変換モジュール122、123、優先順位比較モジュール124、共通割込みハンドラ174、OS間通信機能モジュール175、OSコンテキスト切り替えモジュール176、割込みマスクレベル計算モジュール177は、第一の実施例で説明した内容と同様の構成、処理フローを有する。各オペレーティングシステム内には、優先順位通知モジュール120、121は存在しない。

【0108】

優先順位監視モジュール344は、事務処理OS実行優先順位記憶値342とリアルタイムOS実行優先順位記憶値343を所有する。優先順位監視モジュール344は定期的に起動され、各オペレーティングシステムの実行優先順位340、341を監視する。ここで、記憶している実行優先順位342または343と、現在の実行優先順位340または341が異なっていれば、優先順位が変化した

と認識して、優先順位変換モジュール 122 または 123 に新たな実行優先順位を通知する。

【0109】

優先順位監視モジュール 344 の処理フローを図 24 に示す。第一に、事務処理 OS の実行優先順位 340 を読み出して、これを事務処理 OS 実行優先順位記憶値 342 と比較する（処理 350）。これにより、事務処理 OS の優先順位が変化したか否かを認識できる。優先順位が変化していなければ何も実行せず、処理 353 に移行する。優先順位が変化していれば、事務処理 OS の実行優先順位 340 を事務処理 OS 実行優先順位記憶値 342 に登録し（処理 351）、次に、当該優先順位を事務処理 OS の優先順位変換モジュール 122 に通知する（処理 352）。リアルタイム OS 111 の実行優先順位変化に対しても同様の処理を行う。第一に、リアルタイム OS の実行優先順位 341 とリアルタイム OS 実行優先順位記憶値 343 との比較を行い（処理 353）、優先順位の変化が無ければ処理を終了する。優先順位が変化していれば、リアルタイム OS の実行優先順位 341 をリアルタイム OS 実行優先順位記憶値 343 に登録し（処理 354）、これをリアルタイム OS の優先順位変換モジュール 123 に通知する（処理 355）。

【0110】

なお、優先順位変換モジュールに優先順位を通知した結果、オペレーティングシステムの切り替えが行われると、優先順位監視モジュール 344 に制御が戻ってこないことに注意しなければならない。このため、図 24 の監視方法は、常に事務処理 OS 110 の優先順位変化を優先して監視するものであると言える。両オペレーティングシステムの監視を対等なものとするためには、優先順位監視モジュール 344 の起動回数を計数しておく、

- （１）起動回数が奇数であれば事務処理 OS 110 の優先順位変化を監視し、
- （２）起動回数が偶数であればリアルタイム OS 111 の優先順位変化を監視する

といった処理フローなどに改める必要がある。

【0111】

ここで、優先順位監視モジュール344は、「定期的に」起動され、優先順位の変化を監視することに注意しなければならない。定期的な処理を実現するためには、タイマ割込みによって優先順位監視モジュール344を起動すればよい。ここでは、タイマ割込み、外部割込みを含む全ての割込みが共通割込みハンドラ174で一括処理されることに着目し、共通割込みハンドラ174内に優先順位監視モジュール344を埋め込む方法について説明する。図25が優先順位監視モジュール344を埋め込んだ共通割込みハンドラ174の処理フローである。図中、処理240～248は全て図12に等しい。優先順位監視モジュール344の起動（処理360）はオペレーティングシステム切り替えを行う可能性がある。このため、割込みハンドラの実行（処理245、246）後に埋め込まなければならない。割込みハンドラ152、153が共通割込みハンドラ174に制御を戻すと、自動的に優先順位変化が監視され、必要に応じてオペレーティングシステムの切り替えが行われる。優先順位監視モジュール344がオペレーティングシステムの切り替えを行う必要がないと判断した場合、そのまま共通割込みハンドラ174の処理を終了する。

【0112】

なお、本発明の第二の実施例では、リスケジューリング直後の優先順位変化を優先順位変換モジュールに通知する手段が存在しない。すなわち、一定間隔毎に優先順位の変化をチェックするのであって、優先順位の変化に直ちに反応してオペレーティングシステムを切り替えることにはならない。このため、本発明の第二の実施例は、オペレーティングシステム内部の改造を伴わない計算機を実現できるが、第一の実施例に比べ、切り替え効率的に劣るという特徴がある。

【0113】

本発明の第二の実施例は、両オペレーティングシステムの内部を改造しないという条件を満たすものであるが、例えば、リアルタイムOS111は改造できるが、事務処理OS110を改造することができない、といった条件が発生することもある。この場合、本発明の第三の実施例として、リアルタイムOS111内部に優先順位通知モジュール121を搭載し、優先順位監視モジュール344は

事務処理OS 110の優先順位変化のみを監視するといった計算機を実現することも可能である（図26）。

【0114】

また、リスケジューラ内部に優先順位通知モジュールを組み込むのではなく、各オペレーティングシステムのタイマ割込み処理モジュール内部など、定期的に起動される部分に優先順位通知モジュールを組み込み、優先順位監視モジュール344の代用とすることも可能である。

【0115】

以上説明した本発明を図27に示す車載ナビゲーションシステムに適用した場合、処理時間の長い経路探索タスクが動作している場合でもユーザからの入力を受け付けることができる。つまり、この場合はインタフェースタスクに対して、経路探索タスクよりも高い正規化優先順位を与える。このとき、ユーザがボタンを押してインタフェースタスクが動作すれば、より正規化優先順位の高いタスクを動作させる事務処理OSに切り替えが行われ、結果として、処理の長い経路探索タスクが動作している場合にもインターフェースタスクが実行されることになる。

【0116】

更に、複数のオペレーティングシステムを実行する仮想計算機においても、本発明を適用することにより異なった優先順位体系を有しているオペレーティングシステムを、実行すべきタスクの優先順位にしたがって実行させることができる。

【0117】

以上より、本発明によれば事務処理用OSのユーザインタフェースとリアルタイムOSの信頼性を併せ持つ計算機を構築することができる。

【0118】

前記第一乃至第三の実施例は、全て、オペレーティングシステム切替えプログラム118の内部で優先順位の変換を行う構成を有している。しかしながら、優先順位変換モジュール122、123は、それぞれ、事務処理OS 110、リアルタイムOS 111に個別に対応しているため、これらを、各オペレーティング

システム内部に搭載しても良い。このような変更を加えた第四の実施例を図29に示す。

【0119】

本実施例では、第一に、優先順位変換モジュール122、123が各オペレーティングシステムでの優先順位を正規化優先順位に変換する。これを受けて、優先順位通知モジュール120、121が、優先順位比較モジュール124に対して各オペレーティングシステムの正規化優先順位を通知する。

【0120】

この実施例では、事務処理OS110、リアルタイムOS111とオペレーティングシステム切替えプログラム108とのインタフェースが全て正規化優先順位によって行われる。前記第一乃至第三の実施例では、実行可能タスクが存在する場合には、タスクの優先順位を通知し、実行可能タスクが存在しない場合には、idle状態であることを通知していた。これらの場合、idle状態や、オペレーティングシステム自体が何らかの処理を実行しなければならない状態を、自由に正規化優先順位にマッピングすることができない。本実施例（第四実施例）では、各オペレーティングシステム内で優先順位を正規化優先順位に変換するため、タスク実行中以外の状態も自由に正規化優先順位にマッピングすることが可能となる。

【0121】

【発明の効果】

本発明によれば、複数のオペレーティングシステムを単一プロセッサで動作させる計算機において、各オペレーティングシステムが実行するタスクの優先順位にしたがって、オペレーティングシステムの切り替えを行うことができ、より重要なタスクを優先して実行させることができる。また、本発明では、優先順位変換モジュールによって正規化優先順位に変換し、これによってオペレーティングシステム間の優先順位比較を行うため、異なった優先順位体系を有するオペレーティングシステム同士を実行させる計算機であっても、リアルタイム性・効率性を損なうことがない。

【図面の簡単な説明】

【図 1】

本発明の第一の実施例におけるシステム構成を示す図である。

【図 2】

計算機のハードウェア構成を示す図である。

【図 3】

割込みレベルマスク機能を装備する場合のステータスレジスタを示す図である。

【図 4】

個別割込みマスク機能を装備する場合のステータスレジスタを示す図である。

【図 5】

オペレーティングシステム内部構成を詳細に示す図である。

【図 6】

オペレーティングシステム切り替えプログラム内部構成を詳細に示す第一の図である。

【図 7】

リスケジューラの処理フローを示す図である。

【図 8】

優先順位変換モジュールの処理フローを示す図である。

【図 9】

優先順位比較モジュールの処理フローを示す図である。

【図 1 0】

オペレーティングシステム切り替えプログラム内部構成を詳細に示す第二の図である。

【図 1 1】

OS コンテキスト切り替えモジュールの処理フローを示す図である。

【図 1 2】

共通割込みハンドラの処理フローを示す図である。

【図 1 3】

割込みハンドラの処理フローを示す図である。

【図 1 4】

優先順位上昇方式におけるロック取得モジュールの処理フローを示す図である。

【図 1 5】

優先順位上昇方式におけるロック解放モジュールの処理フローを示す図である。

【図 1 6】

優先順位設定モジュール内部構成を詳細に示す図である。

【図 1 7】

優先順位変換モジュールにおける優先順位逆変換機能の処理フローを示す図である。

【図 1 8】

優先順位設定モジュールの処理フローを示す図である。

【図 1 9】

優先順位継承方式におけるロック取得モジュールの処理フローを示す図である。

【図 2 0】

優先順位継承方式におけるロック解放モジュールの処理フローを示す図である。

【図 2 1】

割込みマスクレベル計算モジュール内部構成を詳細に示す図である。

【図 2 2】

割込みマスクレベル計算モジュールの処理フローを示す図である。

【図 2 3】

本発明の第二の実施例におけるシステム構成を示す図である。

【図 2 4】

優先順位監視モジュールの処理フローを示す図である。

【図 2 5】

優先順位監視モジュールを起動する共通割込みハンドラの処理フローを示す図である。

【図 2 6】

本発明の第三の実施例におけるシステム構成を示す図である。

【図 2 7】

オペレーティングシステム間でのタスク振り分け例を示す図である。

【図 2 8】

共通割込みハンドラの別の構成を示した図である。

【図 2 9】

本発明の第四の実施例におけるシステム構成を示す図である。

【符号の説明】

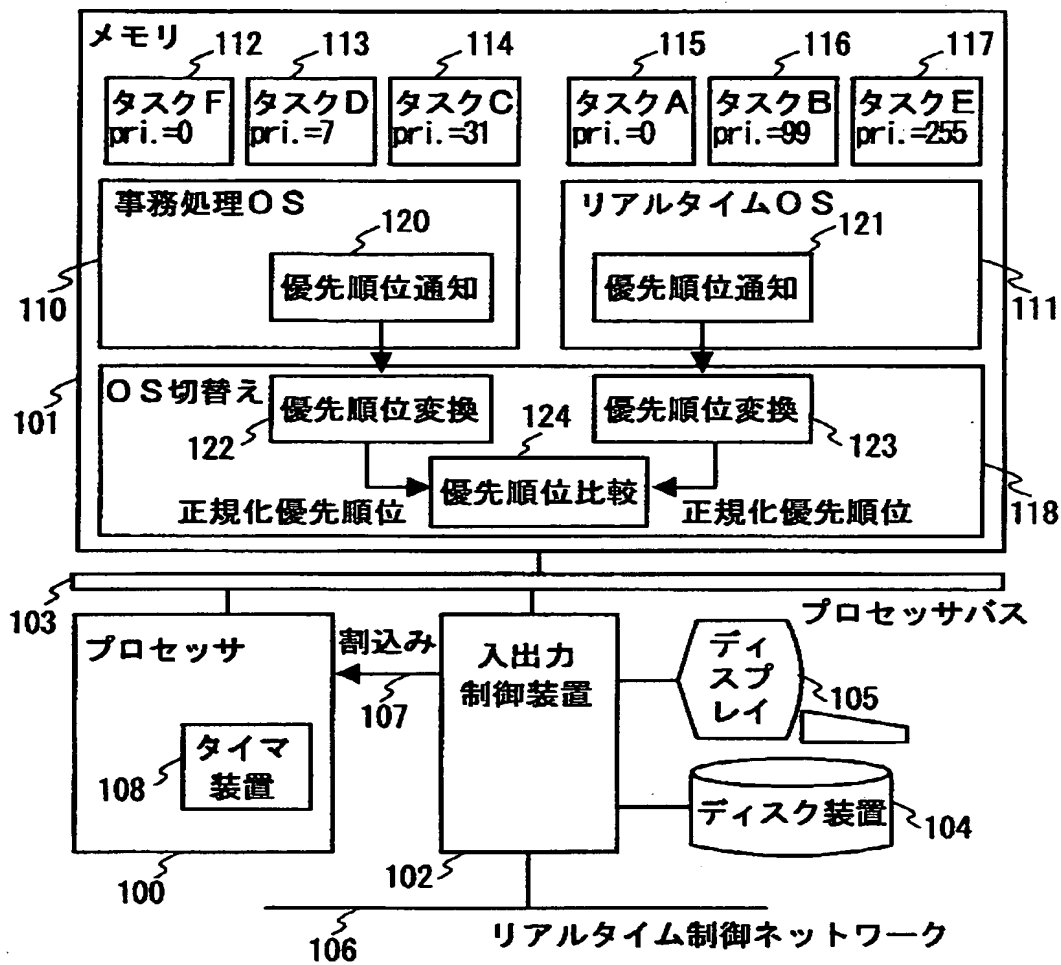
1 0 0 … プロセッサ、1 0 1 … メモリ、1 0 2 … 入出力制御装置、1 0 3 … プロセッサバス、1 0 4 … ディスク装置、1 0 5 … ディスプレイ、1 0 6 … リアルタイム制御ネットワーク、1 0 7 … 割込み信号線、1 0 8 … タイマ装置、1 1 0 … 事務処理 OS、1 1 1 … リアルタイム OS、1 1 2 ~ 1 1 7, 3 7 0 ~ 3 7 3 … タスク、1 1 8 … オペレーティングシステム切り替えプログラム、1 2 0, 1 2 1 … 優先順位通知モジュール、1 2 2, 1 2 3 … 優先順位変換モジュール、1 2 4 … 優先順位比較モジュール、1 3 0 … CPU、1 3 1 … キャッシュメモリ、1 3 2 … 汎用レジスタ、1 3 3 … プログラムカウンタ、1 3 4 … ステータスレジスタ、1 3 5 … 割込みコントローラ、1 3 6 … データバス、1 3 7 … アドレスバス、1 3 8 … 割込み状態信号、1 4 0 … 割込みブロックビット、1 4 1 … 割込みマスクレベルフィールド、1 4 2 … 実行状態レジスタ、1 4 3 … 割込みマスクレジスタ、1 4 4 ~ 1 4 7 … 割込みマスクビット、1 5 0, 1 5 1 … 実行可能待ち行列、1 5 2, 1 5 3 … 割込みハンドラ、1 5 4, 1 5 5 … リスケジューラ、1 5 6, 1 5 7 … システムコールプログラム、1 6 0 ~ 1 6 5 … タスク管理テーブル、1 7 0, 1 7 1 … 優先順位変換テーブル、1 7 2 … 事務処理 OS 正規化優先順位、1 7 3 … リアルタイム OS 正規化優先順位、1 7 4 … 共通割込みハンドラ、1 7 5 … OS 間通信機能モジュール、1 7 6 … OS コンテキスト切り替えモ

ジュール、177…割込みマスクレベル計算モジュール、210…実行OS記憶変数、211…割込み対応テーブル、212…事務処理OS用保存コンテキスト、213…リアルタイムOS用保存コンテキスト、214, 215…割込みスタック、216, 217…割込みスタックポインタ、218…共有メモリ、219…ロック取得モジュール、220…ロック解放モジュール、280, 281…優先順位設定モジュール、282, 283…優先順位逆変換テーブル、320…割込みマスク変換テーブル、340, 341…実行優先順位、342…事務処理OS実行優先順位記憶値、343…リアルタイムOS実行優先順位記憶値、344…優先順位監視モジュール、380…割込み優先順位対応テーブル。

【書類名】 図面

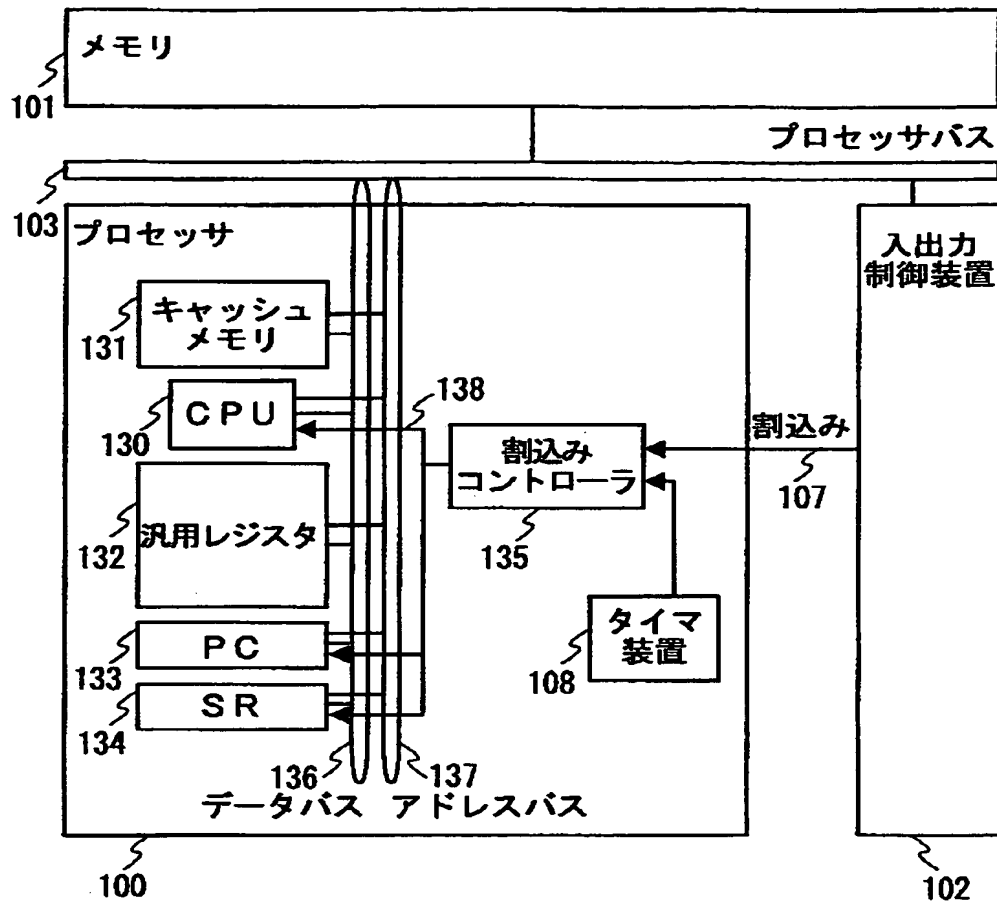
【図 1】

図 1



【図 2】

図 2



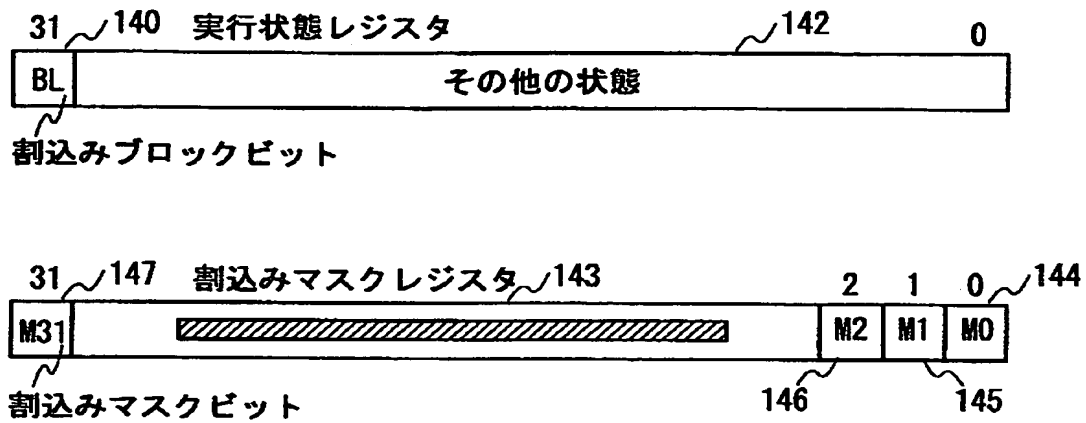
【図 3】

図 3



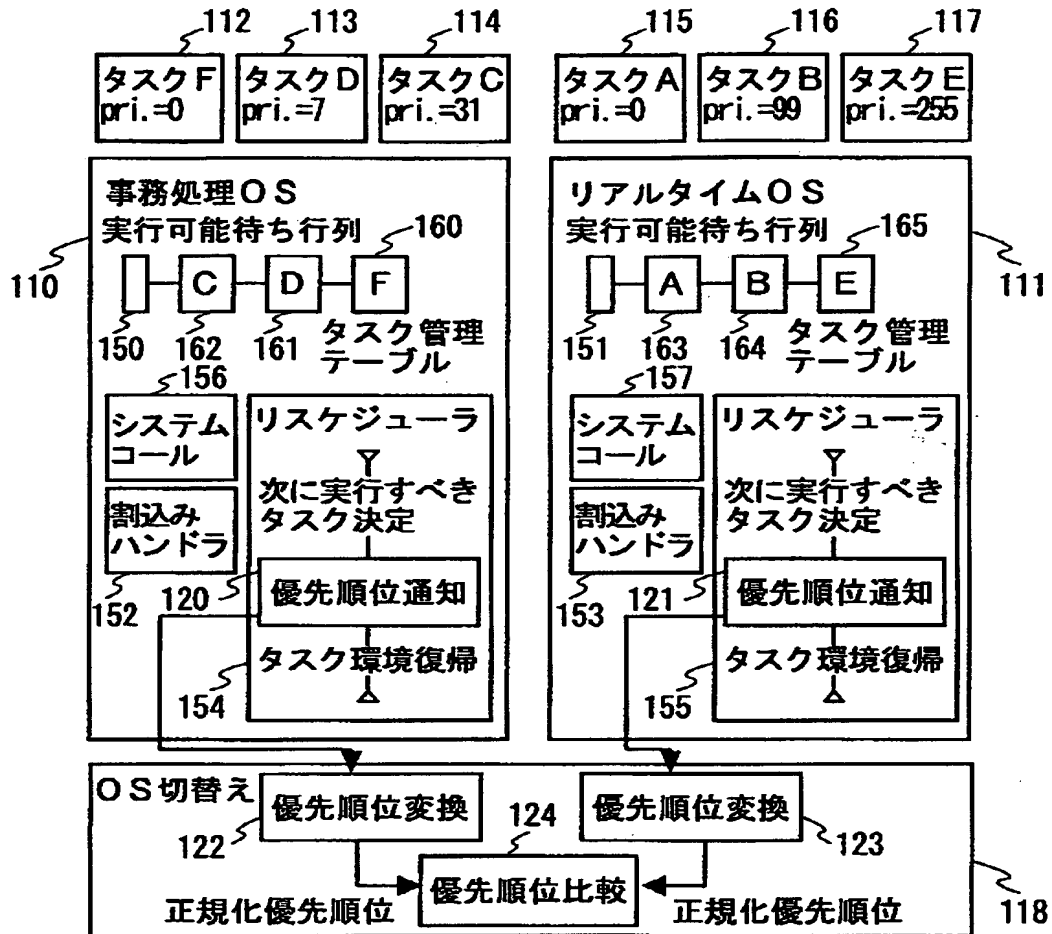
【図 4】

図 4



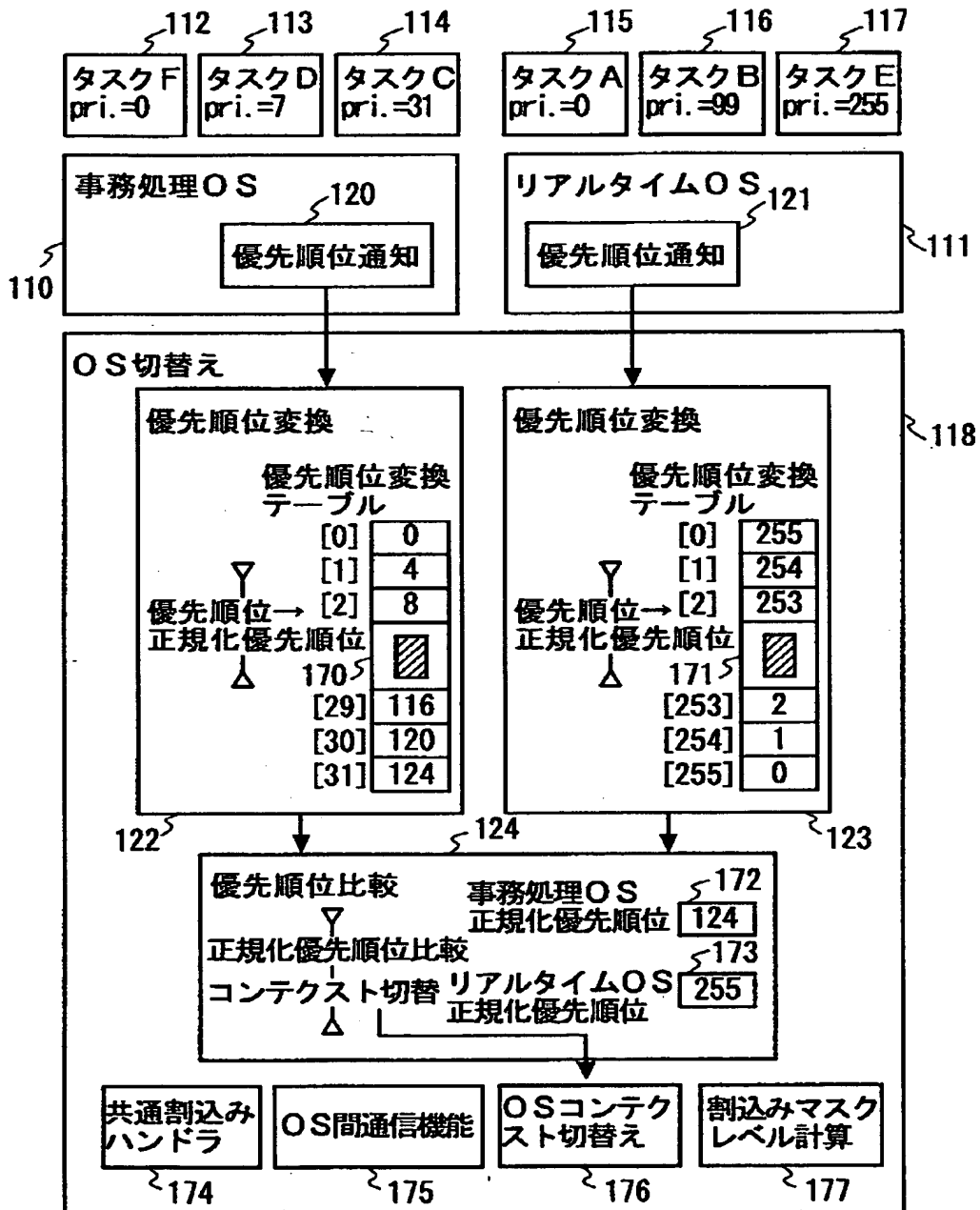
【図 5】

図 5



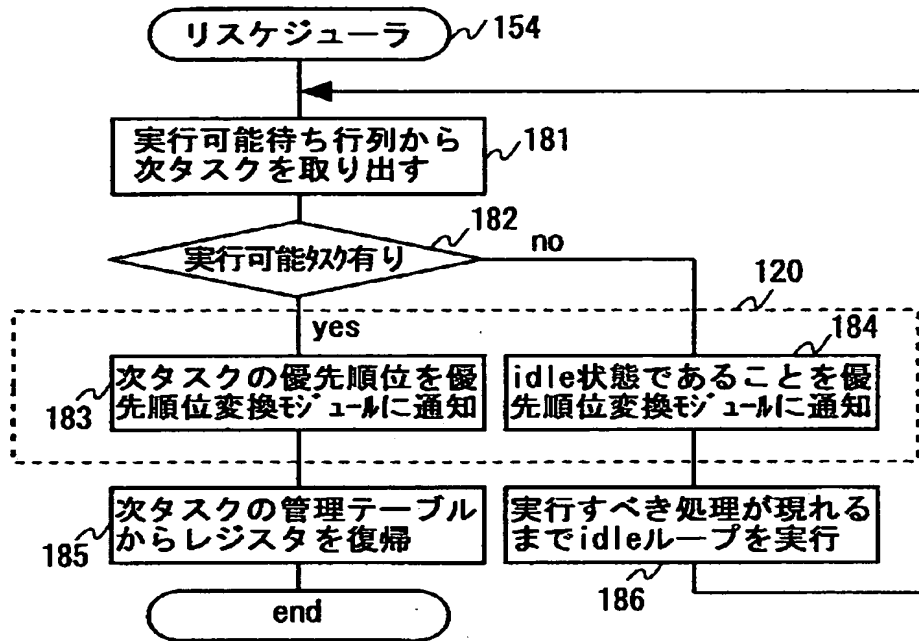
【図 6】

図 6



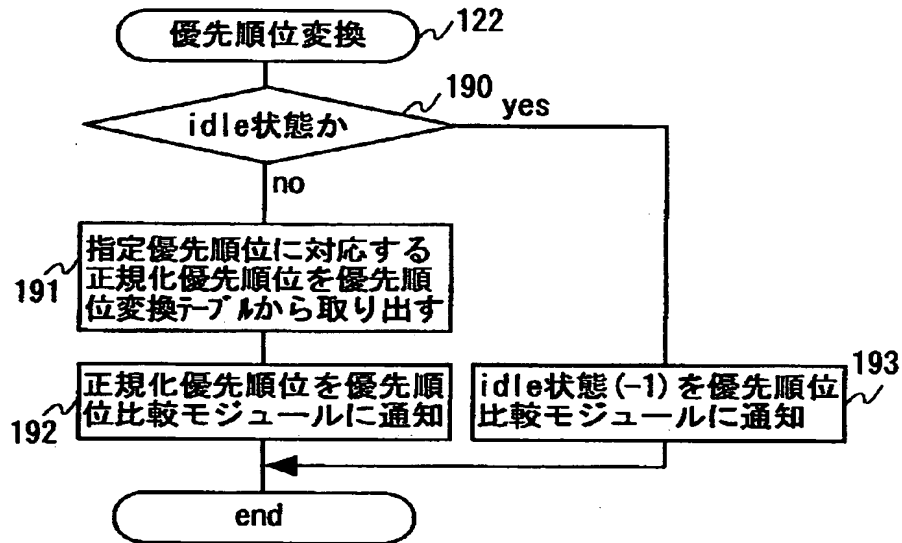
【図 7】

図 7



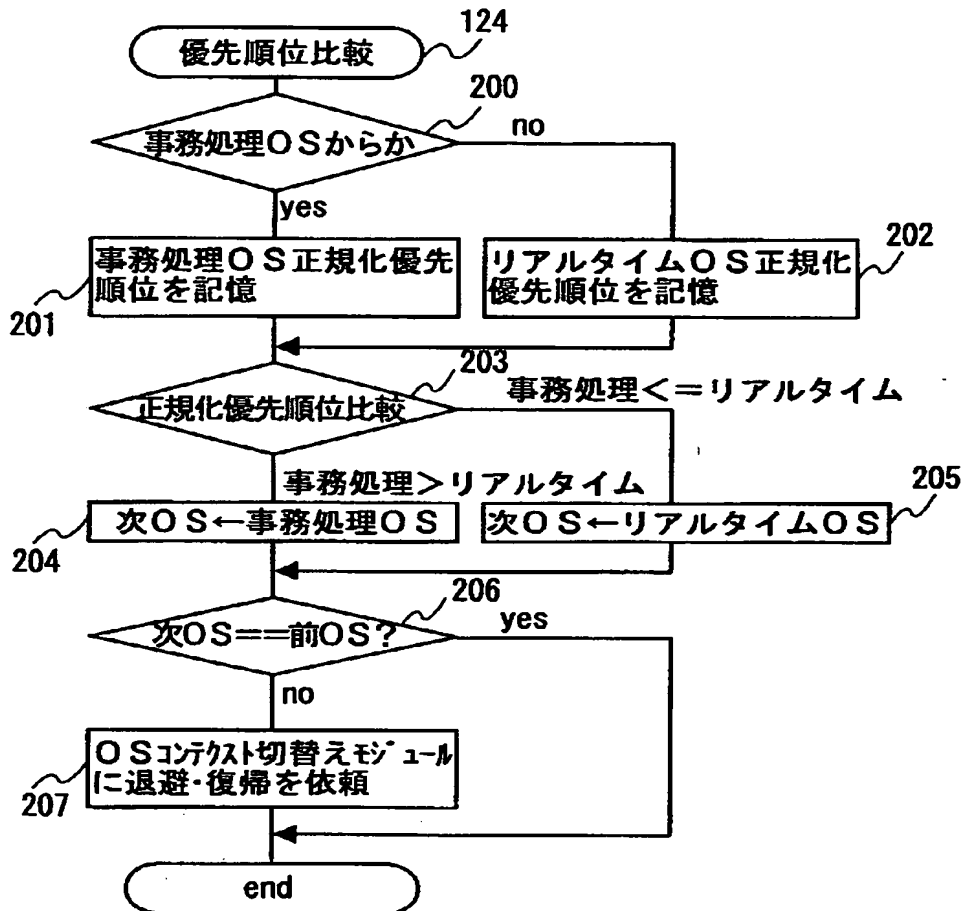
【図 8】

図 8



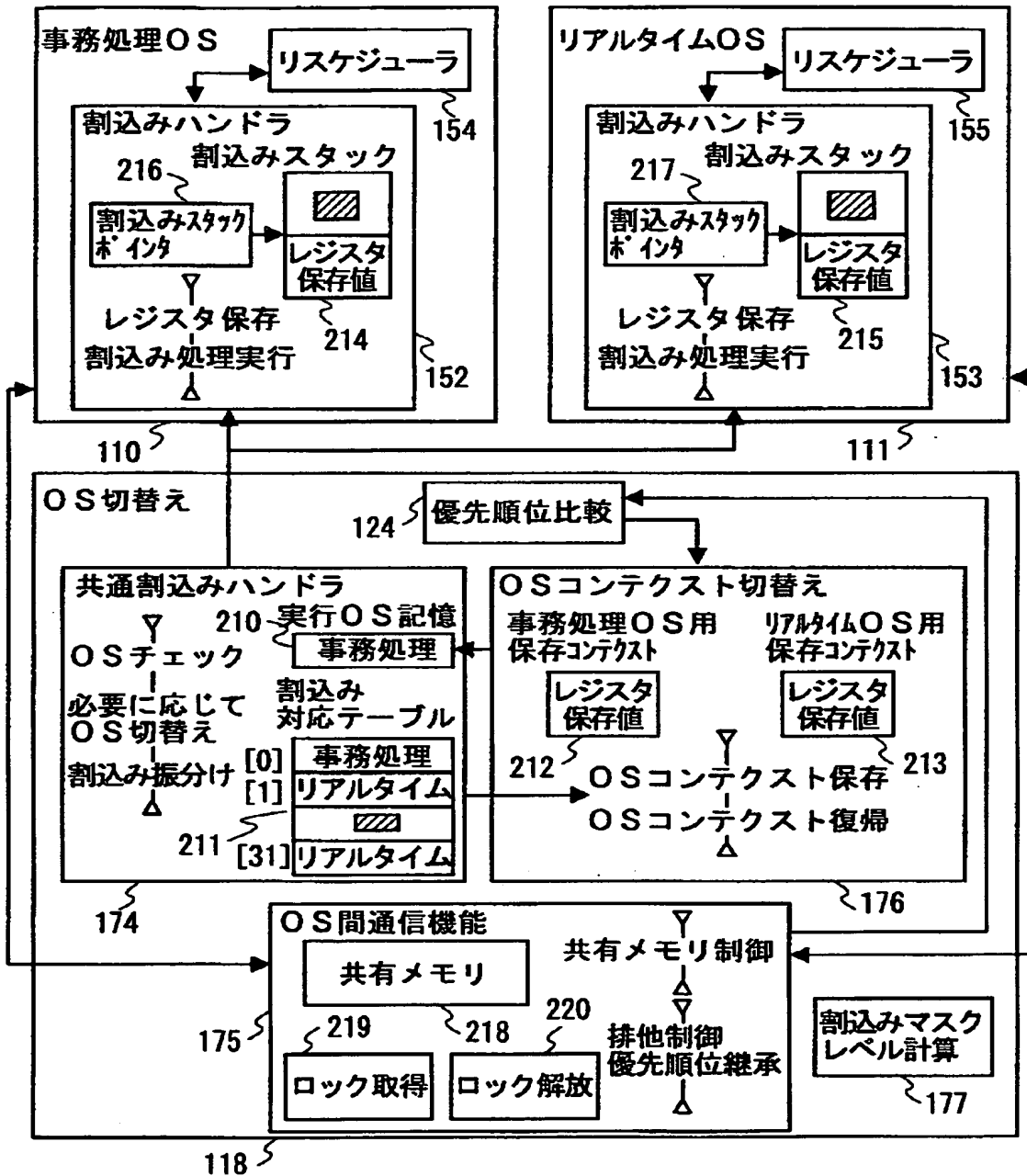
【図 9】

図 9



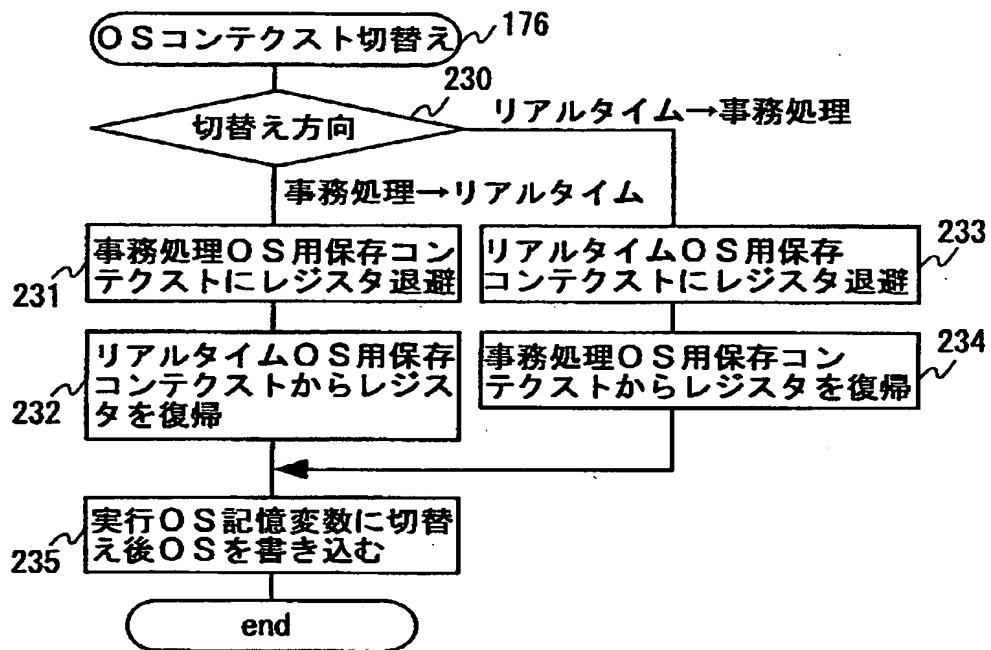
【図 10】

図 10



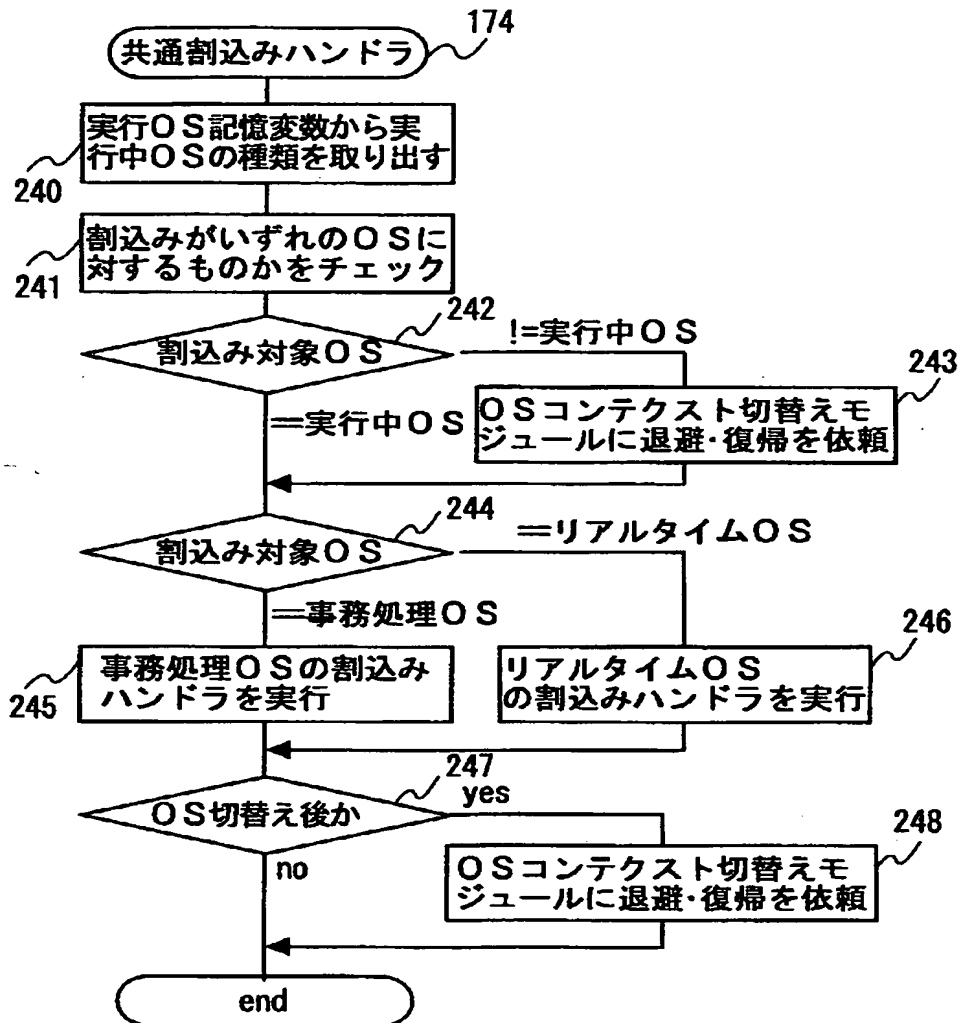
【図 11】

図 11



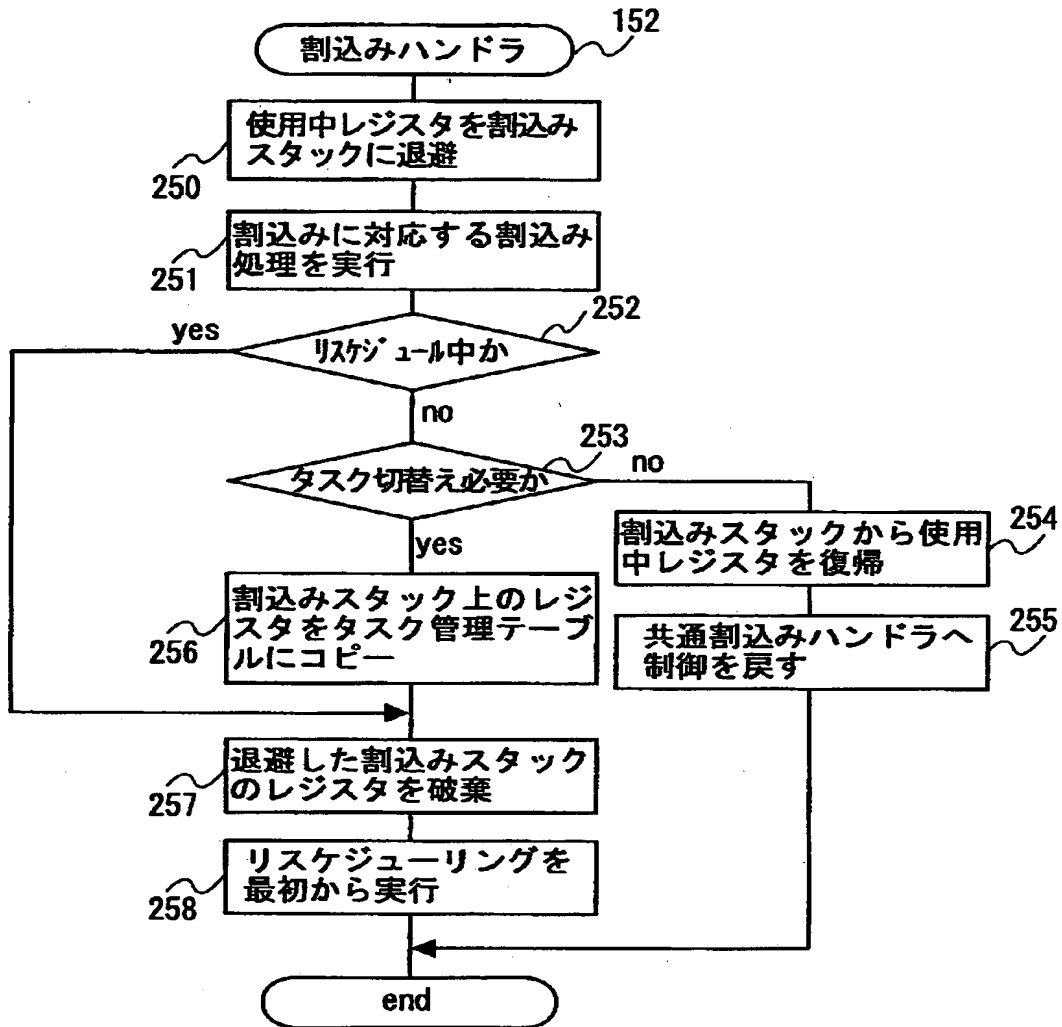
【図 1 2】

図 12



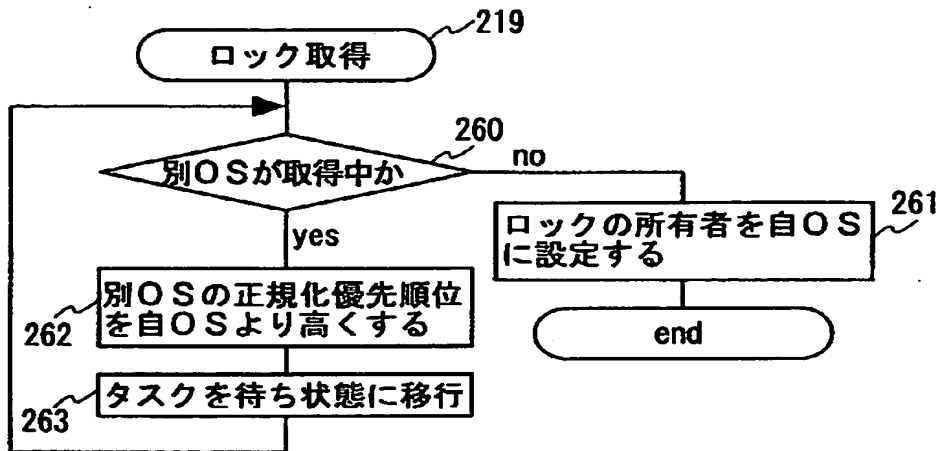
【図 13】

図 13



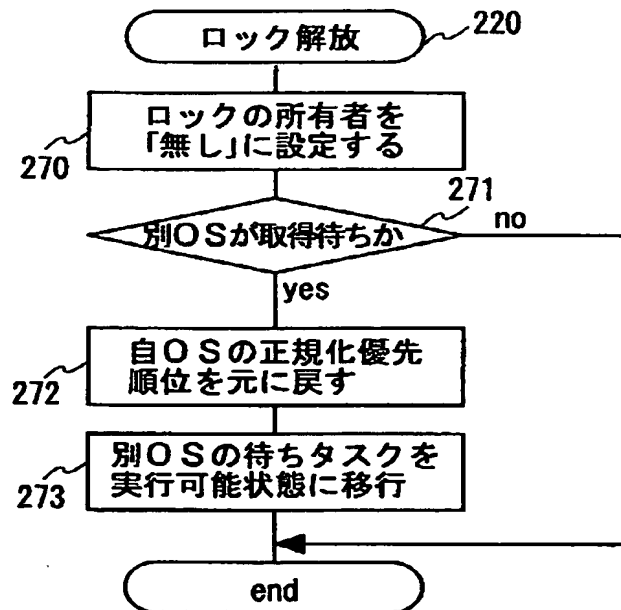
【図 14】

図 14



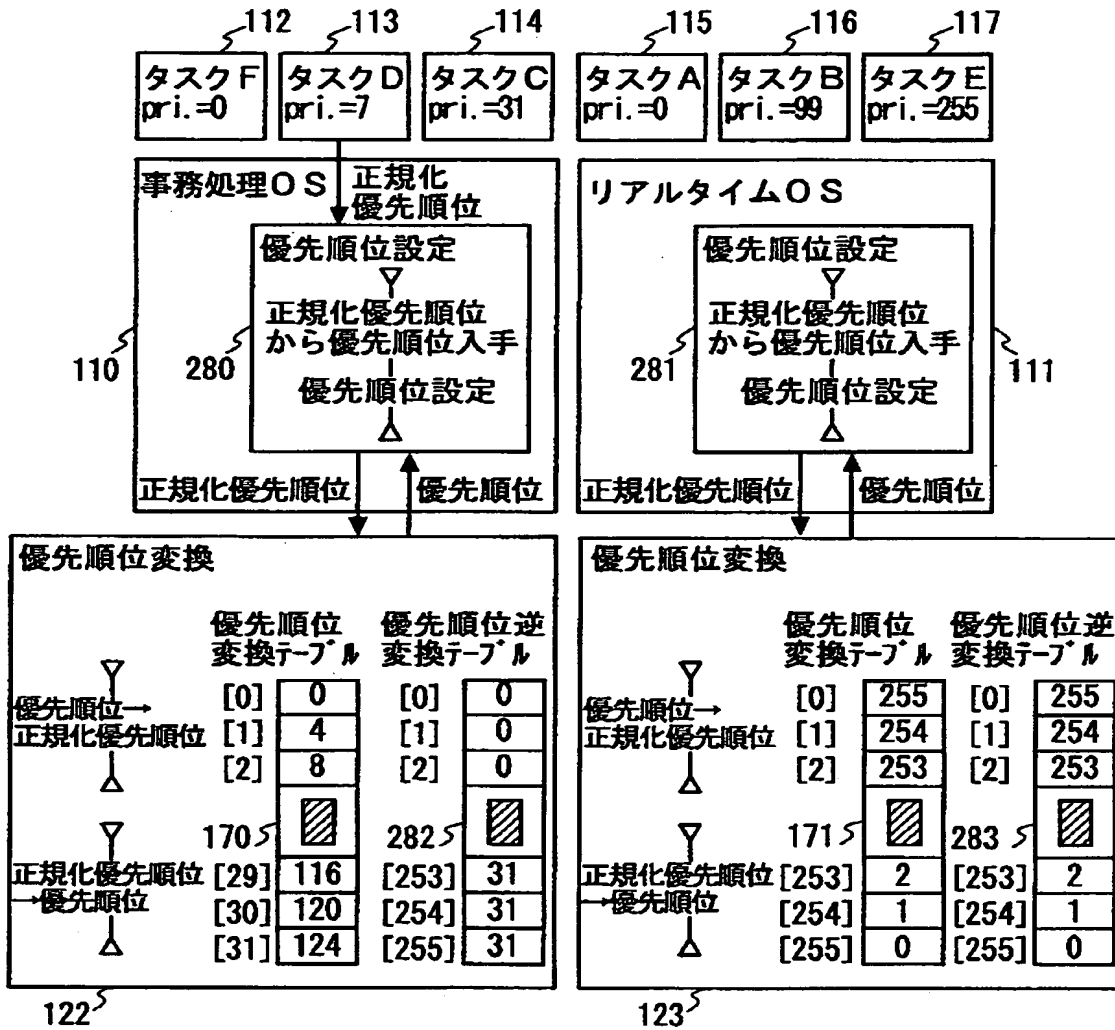
【図 15】

図 15



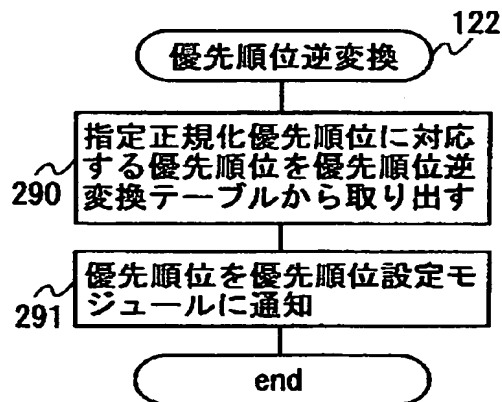
【図 1 6】

図 16



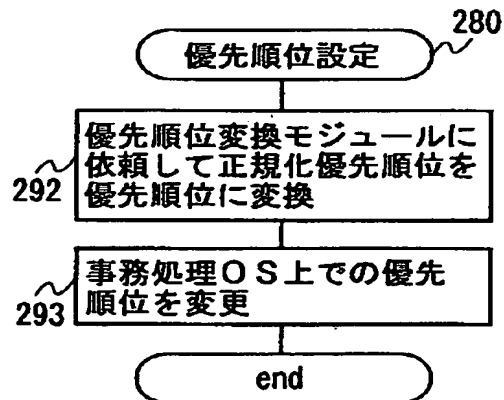
【図 1 7】

図 17



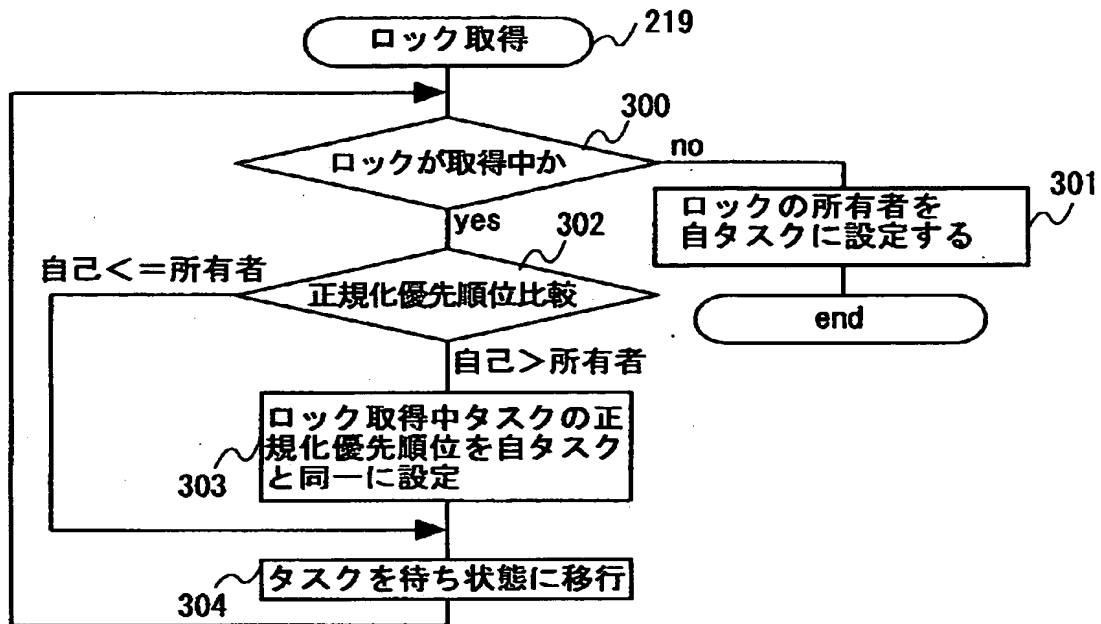
【図 1 8】

図 18



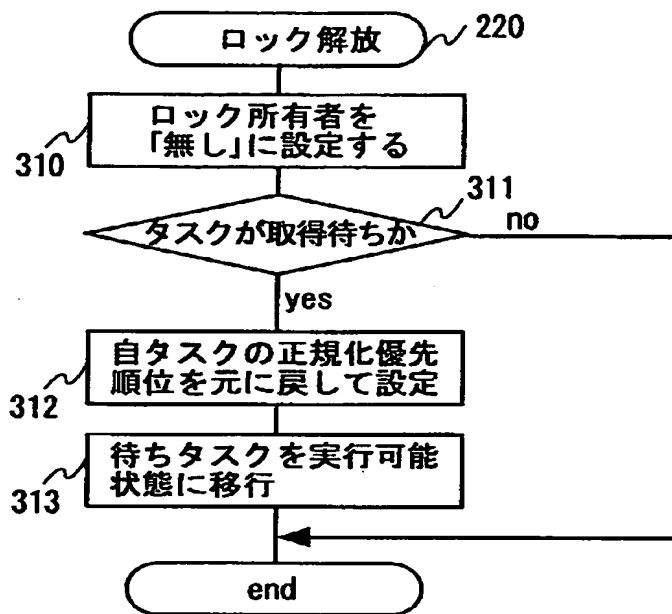
【図 19】

図 19



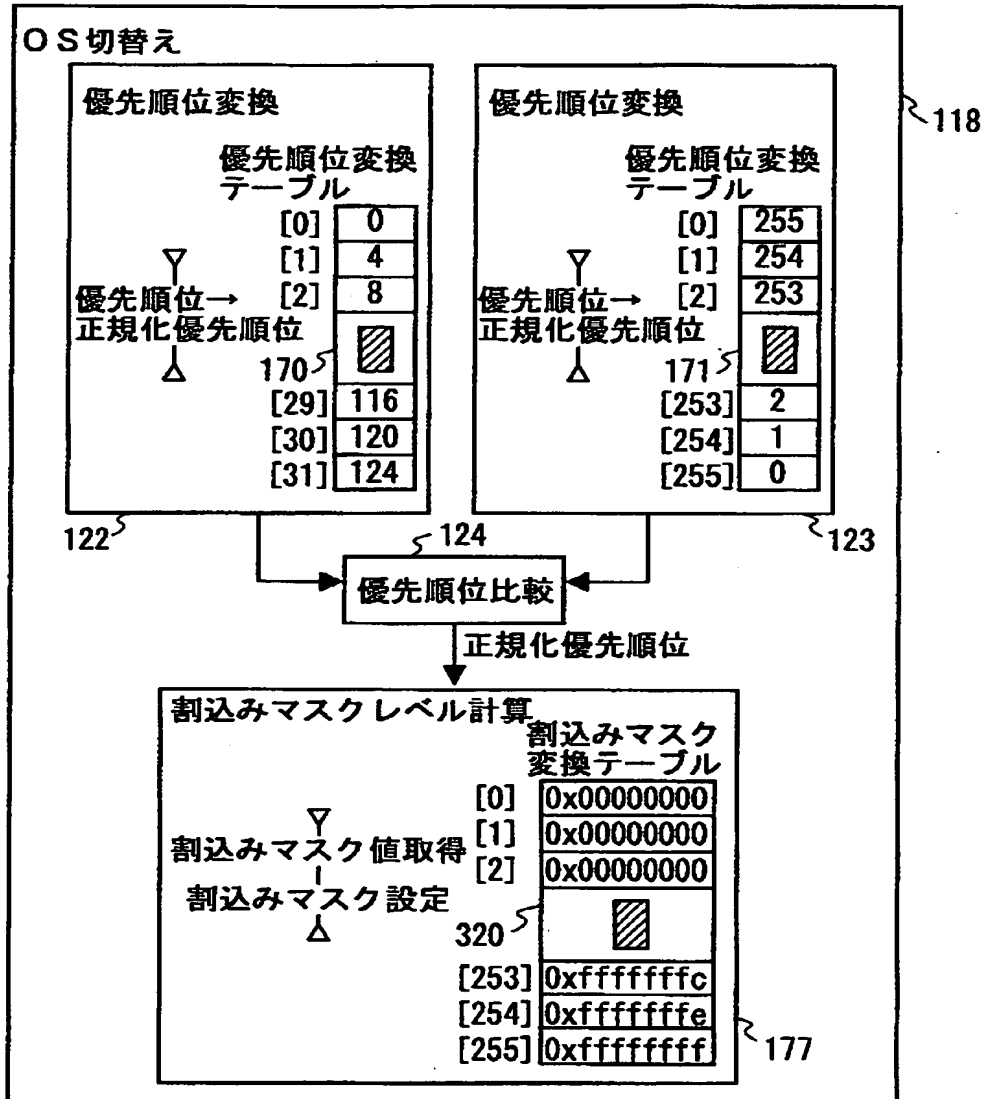
【図 20】

図 20



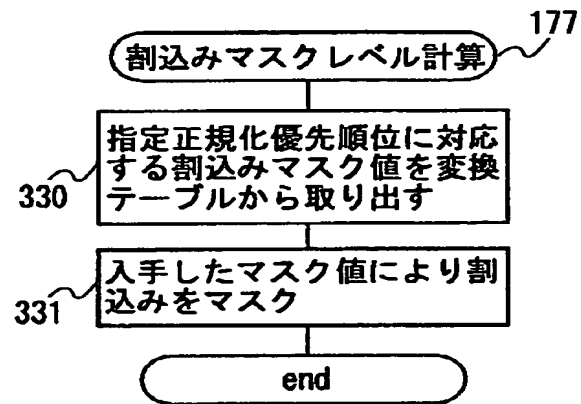
【図 2 1】

図 21



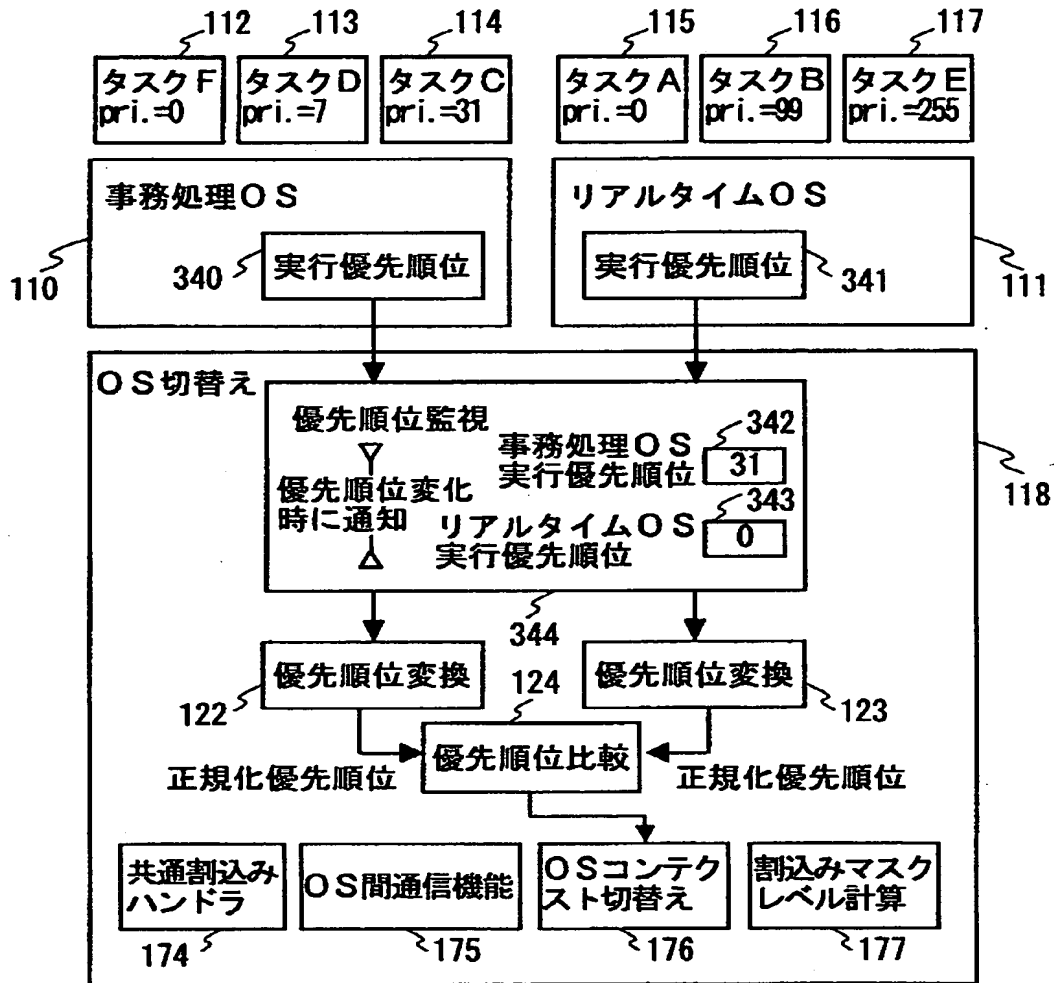
【図 2 2】

図 22



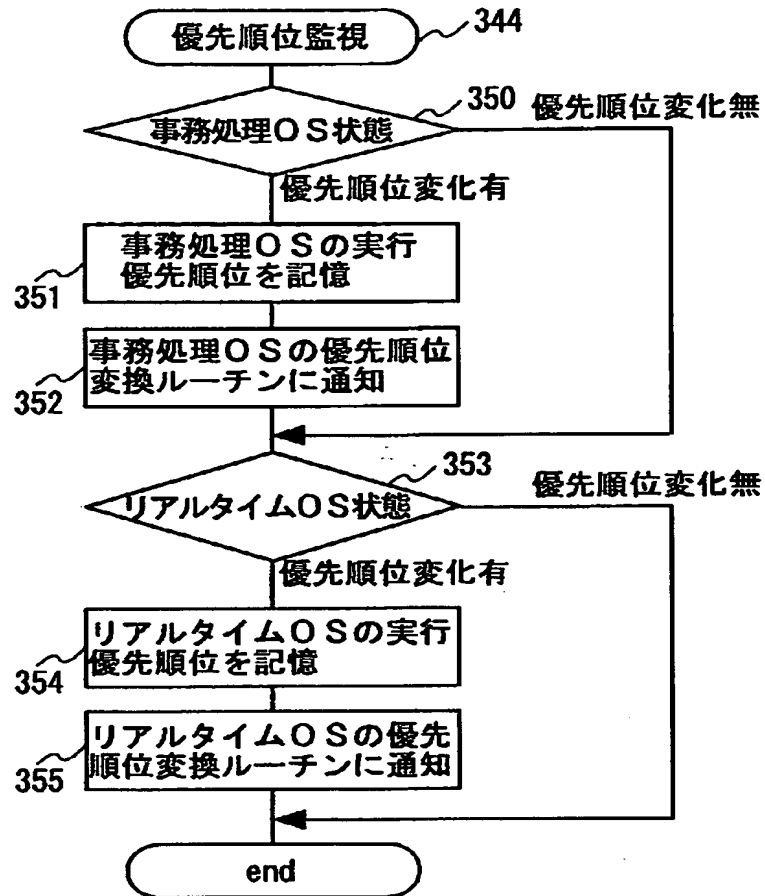
【図 2 3】

図 23



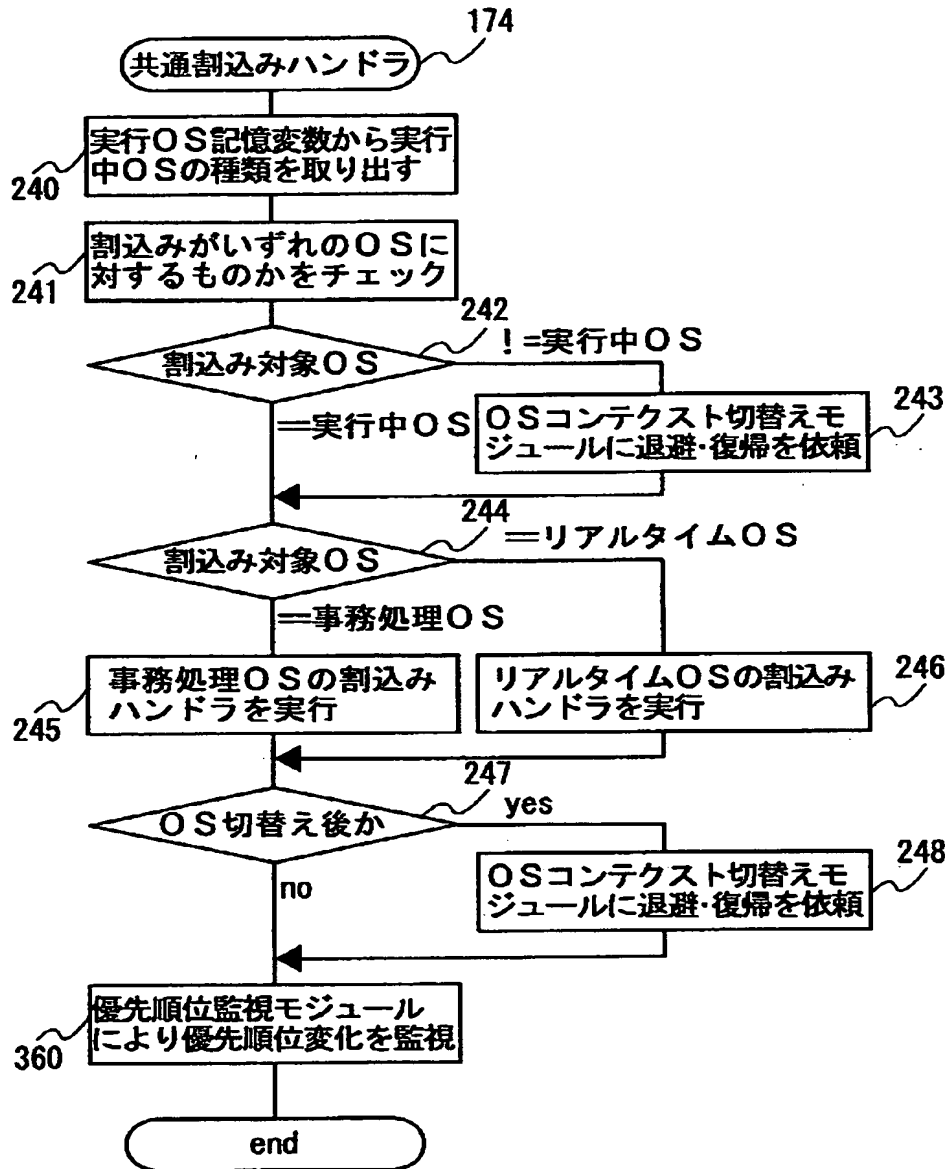
【図 2 4】

図 24



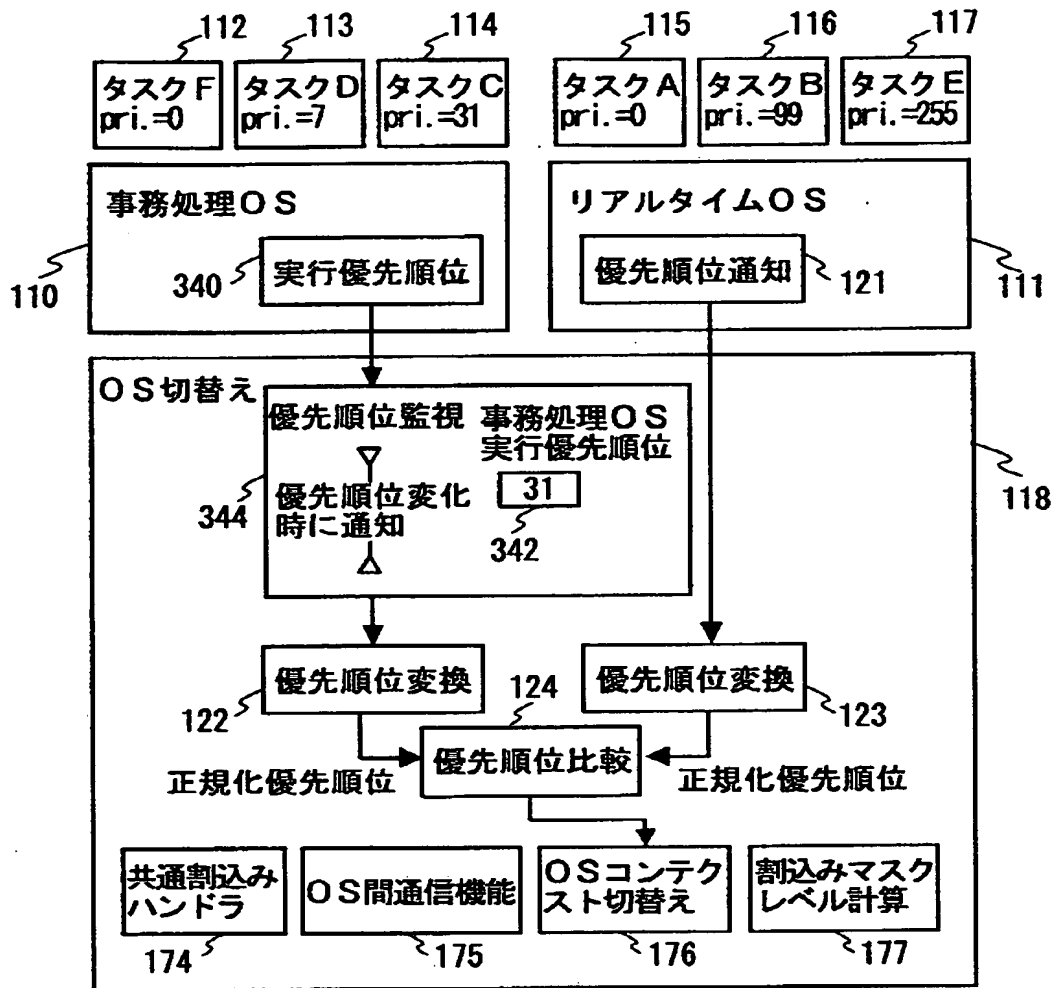
【図 25】

図 25



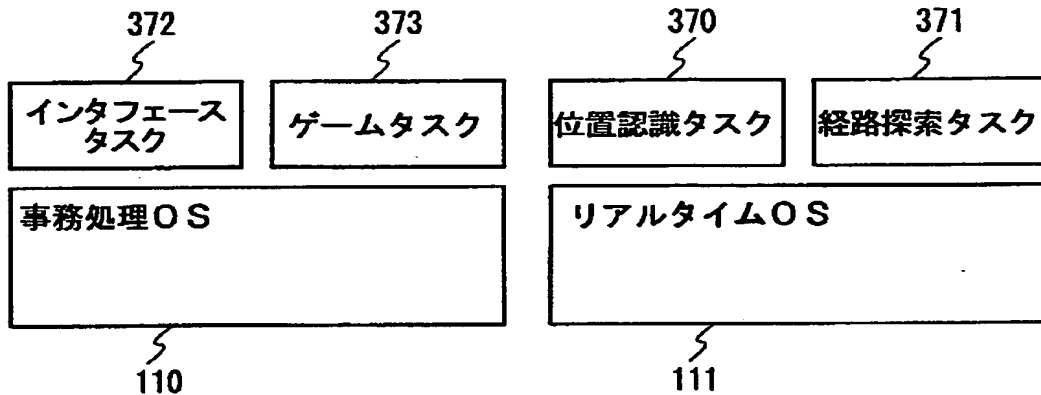
【図 2 6】

図 26



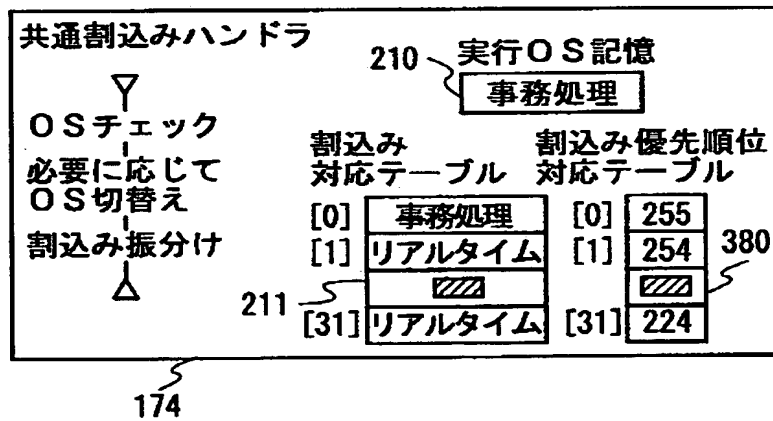
【図 27】

図 27



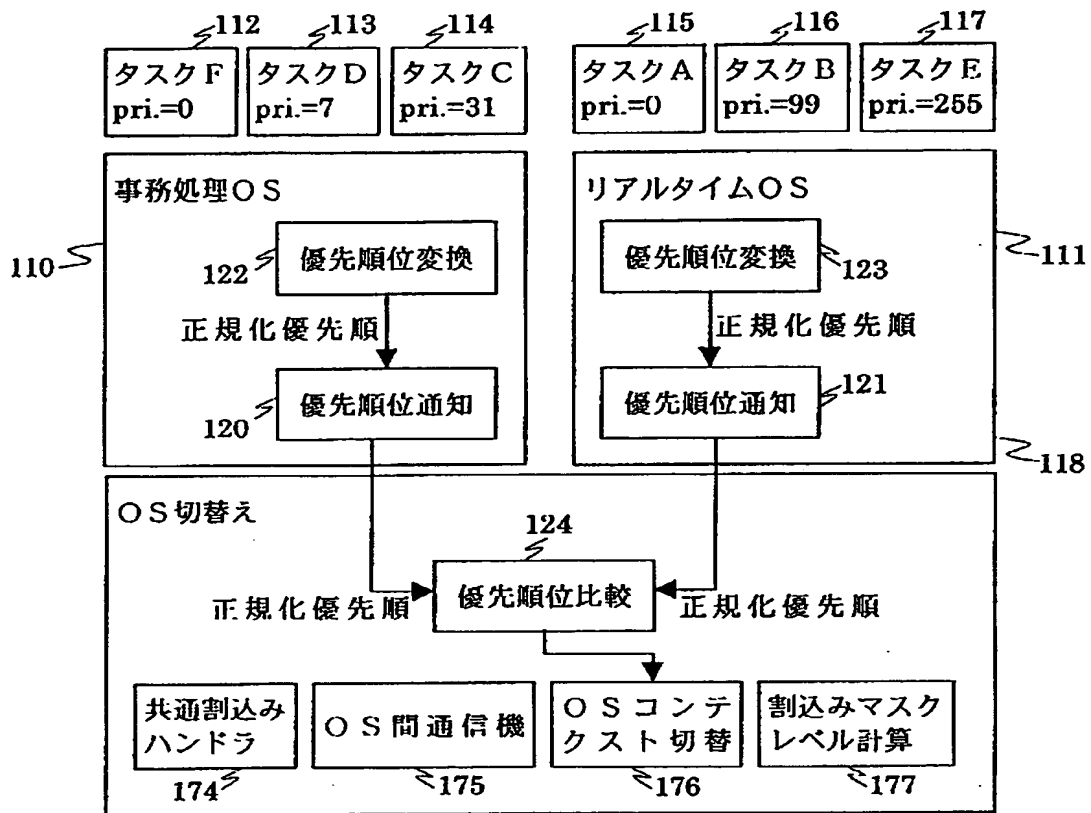
【図 28】

図 28



【図 2 9】

図 29



【書類名】 要約書

【要約】

【課題】

それぞれのオペレーティングシステム上で実行されるタスクの重要性を考慮して、動作させるべきオペレーティングシステムを切り替える。

【解決手段】

複数のプロセスまたはスレッドに割当てられた優先順位にしたがって実行する複数のオペレーティングシステムはそれぞれが実行しているプロセスまたはスレッドの優先順位を通知する優先順位通知処理を有し、それぞれのオペレーティングシステムから通知された優先順位を計算機内で共通の優先順位（正規化優先順位）に変換する優先順位変換処理と、優先順位変換処理によって得られた正規化優先順位を比較して、より高い正規化優先順位を有するオペレーティングシステムを優先的に実行させる優先順位比較処理を実行する。

【選択図】 図 1

認定・付加情報

| | |
|---------|--------------------|
| 特許出願の番号 | 平成11年 特許願 第041032号 |
| 受付番号 | 59900144883 |
| 書類名 | 特許願 |
| 担当官 | 第七担当上席 0096 |
| 作成日 | 平成11年 2月23日 |

<認定情報・付加情報>

| | |
|-------|-------------|
| 【提出日】 | 平成11年 2月19日 |
|-------|-------------|

出 願 人 履 歴 情 報

識別番号 [000005108]

1. 変更年月日 1990年 8月31日

[変更理由] 新規登録

住 所 東京都千代田区神田駿河台4丁目6番地
氏 名 株式会社日立製作所